VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

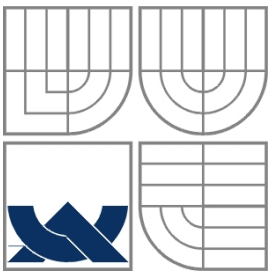
DESKOVÁ HRA VLÁDCI PODZEMÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

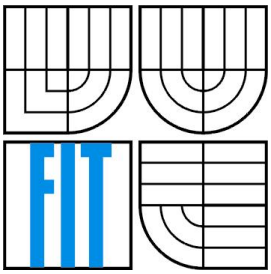
AUTOR PRÁCE
AUTHOR

LUKÁŠ KAWULOK

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

DESKOVÁ HRA VLÁDCI PODZEMÍ

DUNGEON LORDS BOARD GAME

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LUKÁŠ KAWULOK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL DOLEŽAL

BRNO 2012

Abstrakt

V této práci se zabývám převodem deskové hry Vládci podzemí na PC. Počítačová verze hry se bude řídit stejnými pravidly, jako desková hra. Výsledné řešení umožňuje síťovou hru, možnost uložení aktuálního stavu hry, zpětné nahrání a pokračování ve hře. V době tvorby tohoto projektu, je k dostání tato hra pouze v deskové podobě.

V dokumentaci uvádím postup, jakým bylo řešení vedeno a také pravidla hry. Na konci je uveden průběh testování hry v praxi a odhalení nedostatků a následná oprava nalezených chyb. Projekt byl vytvářen v jazyce C# v prostředí Microsoft Visual Studio 2010, k vytvoření GUI je využito techniky WPF.

Abstract

In this work I am describing transformation board game Dungeon lords into computer game. Computer version of the game will have the same rules as board version.Final build of the game allows network playing, saving of game and loading of the saved game. During development of the game is aviable only board version.

In documentation I describe steps of developing and also rules of the game. In the end is described the way of testing and also revieling of bugs and their repair. Projet was created in C# language in MS Visual Studio 2010. To creation of GUI is used WDF technique.

Klíčová slova

Vládci podzemí, C#, GUI, WPF, XAML, síťová komunikace, NAT, Hamachi, pravidla, rozhodování, vyhodnocování

Keywords

Dungeon lords, C#, GUI, WPF, XAML, networking, NAT, Hamachi, rules, decision, evaluation

Citace

Lukáš Kawulok: Desková hra Vládci podzemí, bakalářská práce, Brno, FIT VUT v Brně, 2012

Desková hra Vládci podzemí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Doležala. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Kawulok
Datum (18. 5. 2012)

Poděkování

Na tomto místě bych chtěl poděkovat svému vedoucímu Ing. Michalovi Doležalovi, za jeho podporu a cenné rady při implementaci této práce.
Také bych chtěl poděkovat Bc. Ondrovi Polesnému, ke kterému jsem při vypracovávaní této práce chodíval na cvičení předmětu IW5 a dával mi užitečné rady ohledně jazyka C#.

Obsah

1 Úvod

Desková hra Vládci podzemí je určena pro náročnější hráče, kteří nemají rádi stereotypy. Hra zlepšuje strategické dovednosti a schopnost odhadování tahů protihráčů. Vysvětlení pravidel hry trvá přibližně 30 minut. Samotné studování pravidel z manuálu trvá okolo 4 hodin. K plnému pochopení hry je potřeba si minimálně jednou na nečisto zkusit hru zahrát. Tato hra je k dostání v deskové podobě na naších trhu v plné lokalizaci, ale také v zahraničí v angličtině. Autorem této hry je Vladimír Chvátil, který již má na svém kontu více povedených her.

Tuto hru jsem se rozhodl převést pro hru na počítač z důvodu její složitosti a také jejímu velice podařenému grafickému zpracování v původní podobě. Samotná možnost nebýt vázaný na nutnost scházení hráčů ke hře v deskové podobě je velmi uvolňující. Také se může hodit ukládání hry, při nutném přerušení a následném nahrání hry při shledání všech hráčů na počítačích.

V následující kapitole si shrneme pravidla hry, která vycházejí z originálních pravidel[1]. Poté navrhneme implementaci a podle návrhu, se začneme zabývat samotnou implementaci hry na PC v jazyku C# ve Visual Studiu společnosti Microsoft. Bude zde také důkladně popsáno rozdělení řešení na více projektů a jejich vzájemná komunikace. Dále bude popsána řídicí logika hry a také komunikace mezi klientem a serverem, která je využita při hře po síti. Závěr této práce patří výsledkům testovaní této hry, testování má za úkol zjištění a následné opravení logických chyb. Test byl proveden skrze deset dobrovolníků. Tito „testeři“ již někdy hru hráli a tedy znají deskovou hru Vládci podzemí.

2 Seznámení s hrou

Desková hra Vládci podzemí byla vydána roku 2009 firmou Mindok. Autorem této nikterak jednoduché hry je Vladimír Chvátil. Hra nesklidila úspěch pouze u nás, dostala se i za naše hranice, kde o ní byl velký zájem. Hra je určena pro minimálně 2 a maximálně 4 hráče. Cíl hry je odhalen v podstatě již v názvu. Úkolem každého hráče je tedy stát se vládcem podzemí.

2.1 Pravidla hry

Hra se skládá ze dvou základních části. Jedná se o část budovací a bojovou. Společně tyto části tvoří jeden herní rok, který se ve hře opakuje dvakrát.

2.1.1 Budovací část

Každá hra začíná právě touto částí. Budovací část je rozdělena na 4 kola. Tato kola jsou představována ročními obdobími a tvoří jeden rok hry. Každé kolo je sestaveno z určitých fází, které jsou vyznačeny na plánu roku.

Úkolem budovací části je připravit se co nejlépe na bojovou část. Hráč se snaží si nasbírat co nejvíce suroviny, zlata, impů, pastí, chodeb a monster. Monstra a pasti využije při boji proti hrdinům. Budovatelská část končí rozdáním třetího hrdiny.

Příprava kola

Na centrální plán se vyloží tři příšery a dvě místnosti, dále se táhnou hrdinové, kteří se vyloží na plán roku. Tito hrdinové jsou seřazeni dle jejich síly. V neposlední řadě se také odkryje žeton události.

Hraní příkazů

Každý hráč si tajně na svou desku připraví 3 příkazy. Postupně od začínajícího hráče se odkrývají příkazy. Při odkrytí karty hráč položí figurku svého služebníka na odpovídající pole a po odkrytí všech karet se tyto příkazy hrají. Na každém poli je vyznačeno, co musí hráč provést, aby mohl svůj příkaz zahrát. V případě že hraje maximální počet hráčů a všichni zahrají stejný příkaz, na jednoho z nich nezbude místo a svůj tah tedy nemůže zahrát a svého služebníka vrátí na danou kartu.

Fáze výrobních místností a stažení karet

V této fázi mohou hráči využít své místnosti. K tomuto procesu využívají impy. Dalším krokem nutným provést je stažení příkazových karet. Hráč stáhne obě karty z pozice zablokovaných příkazů.

Ze tří karet si vezme první v pořadí, nebo kartu, na které se nachází služebník zpět do své ruky a zbylé 2 posune na zablokované pozice.

Vyhodnocení události

Budovací část ztěžují události. Mezi události, které mohou být odkryty, patří: platba daní za počet chodeb a místností, opětovné placení za již pořízené příšery a zvláštní událost. V případě nezaplacení daně za chodby a místnosti se hráči přidělí červený puntík, který na konci hry odečítá 3 body. Při neschopnosti znovu zaplatit cenu příšery se hráči tato příšera odebírá. Zvláštní událost se vyhodnocuje dle karty.

Rozdělení hrdinů

Nejslabší hrdina se přidělí hráči, který se nachází na nejnižší pozici na stupnici zla a naopak nejsilnější protivník se přidělí hráči, který se nachází nejvýše. Hrdinové mají různé charaktery. Ve hře se objevuje bojovník, který musí stát vždy nejblíže vchodu do dungeonu. Dále se mezi protivníky objevuje kněz, který dokáže léčit zranění své družiny. V případě, že hráč má mezi svými protivníky čaroděje, platí na něj efekty bojových karet. Posledním druhem hrdiny je zloděj, jež odhaluje pasti.

Konec kola

Impové se stáhnou z dungeonu a mění se začínající hráč.

2.1.2 Bojová část

Při dokončení budovací části se plán roku otočí na druhou stranu a umístí se na něj bojové karty. Tato část hry se skládá minimálně z jednoho kola a maximálně ze čtyř kol. Dříve než se odkryje bojová karta pro každé kolo, si hráč naplánuje svůj tah. Na boj v chodbě může nasadit jednu příšeru a ducha, navíc může také nastražit jednu past. Při boji v místnosti může nachystat dvě příšery a také jednu past. Při nachystání pasti musí ovšem přidat jedno zlato. Není povinností nasadit do boje monstrum ani past. Hráč si také sám určí, kde se boj bude odehrávat. Musí se však jednat o nedobyté pole, které je nejblíže k vchodu. Toto vybrané pole označí figurkou služebníka. Samotný postup boje je zakreslen na každé hráčské desce.

Bojové karty mají efekt pouze v případě, že se v družině nachází kouzelník. Na této kartě je také označeno, zda se jedná o rychlé kouzlo, či pomalé. Otočení karty je ale důležité pro všechny hráče. Na každé kartě je zobrazen počet zranění, které družina získá při dobývání.

Cílem hrdinů je dobýt co nejvíce chodeb či místností hráčova dungeonu. Při každém dobytém poli se hráč posune na stupnici zla k dobru. Hrdina je vyřazen z boje v případě, že ztratí všechny své životy. Počet životů je zakreslen na žetonu hrdiny a zranění od monster či pastí jsou znázorněna

červenými kostkami. Bojová část končí v případě, že hráč porazí všechny hrdiny a zajme je do vězení. Může ovšem nastat situace, kdy uplyne 4. kolo, ale hrdinové jsou stále ve hře. V této chvíli hrdinové unikli a hráč přijde o několik bodů při konečném sčítání. Poslední možnou situací ukončení boje je případ, kdy je dungeon zcela dobyt. V této chvíli hráč odhodí jednoho z již zajatých hrdinů.

2.1.3 Druhý rok

Po skončení prvního roku hra nekončí. Začíná druhý rok hry. Hráči pokračují dále se stejným dungeonem. Dobyté chodby a místnosti z boje jsou pro druhý rok hry zcela nevyužitelné a nedají se ani nijak obnovit. Pro druhý rok se vymění bojové karty, na centrální plán se dají monstra určená pro tento rok a také se ztíží hrdinové. Hlavní kostra hry se ovšem ani v druhém roce nemění. Začíná se tedy budovací částí a navazuje na ni část bojová.

2.1.4 Závěrečné bodování

Na bojové straně plánu roku je počítadlo bodů. 2 body se přidělují za každou nedobytou místnost v dungeonu a také za každého zajatého hrdinu. Za každou příšeru se přičítá jeden bod. Další body mohou hráči přibýt díky speciálním místnostem. Naopak hráč přijde o 2 body za každé dobyté pole a také se hráči mohou odečíst 3 body za nezaplacenou daň. Hráč může dále získat body za udělené tituly, jedná se o 7 titulů. 3 body se přičítají za exkluzivní titul a za sdílený titul s jiným hráčem se přičítají body 2. Tyto tituly jsou udělovány za nejvyšší dosažené hodnoty určitých aspektů. Jedná se o tyto aspekty: počet místností, příšer, chodeb, zlata, jídla; nejvyšší pozice na stupnici zla a nejméně dobytých polí dungeonu.

2.1.5 Hra ve dvou, či třech hráčích

V jiném než maximálním počtu hráčů se pravidla hry lehce mění. Ve dvou hráčích se na začátku hry položí na stupnici zla také jedna kostka nehrající barvy. Ve fázi hrání příkazů obsluhují nehrající desky hráči ve hře. Při každém počtu hráčů se tedy musí využít všech 12 služebníků. I nehrající barva může zablokovat příkaz hrajícímu hráči. Služebníci nehrající barvy se přidělují při 3 hráčích vždy na prostřední pole. Při 2 hráčích se první 2 příkazy přidělí také na prostřední pozici, ale třetí příkaz umístí na první pole. Při rozdělování hrdinů jsou také nehrající barvy potřebné. Před rozdáním hrdinů se kostka barvy, která není ve hře, posune o jedno ke zlu a přidělí se jí patřičný hrdina. Tento hrdina se ale rovnou odloží na plán vzdálených krajů. Na začátku druhého roku se posune kostka nehrající barvy na stupnici zla o dva, směrem k dobru. Poslední změnou je přidělování bodů za tituly. Ve hře dvou hráčů se za tento titul přidělují pouze 2 body a za sdílený titul se přiděluje pouze jeden bod.

2.2 Herní plány

Pro zpřehlednění celé hry slouží několik plánů, které se využívají ve hře a každý z nich plní specifický účel.

2.2.1 Centrální plán

Na tomto plánu jsou zobrazeny nejdůležitější prvky, které se využijí při budovací části. Jedná se o: nákup jídla; posun na stupnici zla směrem dolů; těžbu zlata; kopání chodeb; nákup impů; získávání pastí, monster a místností. Součástí této desky je také stupnice zla. Na tuto stupnici dá každý hráč kostku své barvy.

Obr. 1: Centrální plán.

2.2.2 Plán roku

Tento plán slouží k určování fáze hry. Lze také vidět kolik kol zbývá do ukončení budovací části. Na tomto pláně mají své místo také 3 události, které mají za úkol ztížit hráčům budování. Tyto události jsou označeny otazníkem. Plán roku má také své místo pro hrdiny, kteří se v pozdější fázi hry přidělí určeným hráčům. Podle pozice na stupnici zla jsou hráčům přidělováni hrdinové. Silnější hrdinové připadnou „více zlým“ hráčům. Při překročení určité hranice zla míří k hráči paladin, který je nejtěžší možný protivník. Z druhé strany desky je nakreslen plán bojové části hry.

Obr. 2: Plán roku.

2.2.3 Vzdálené kraje

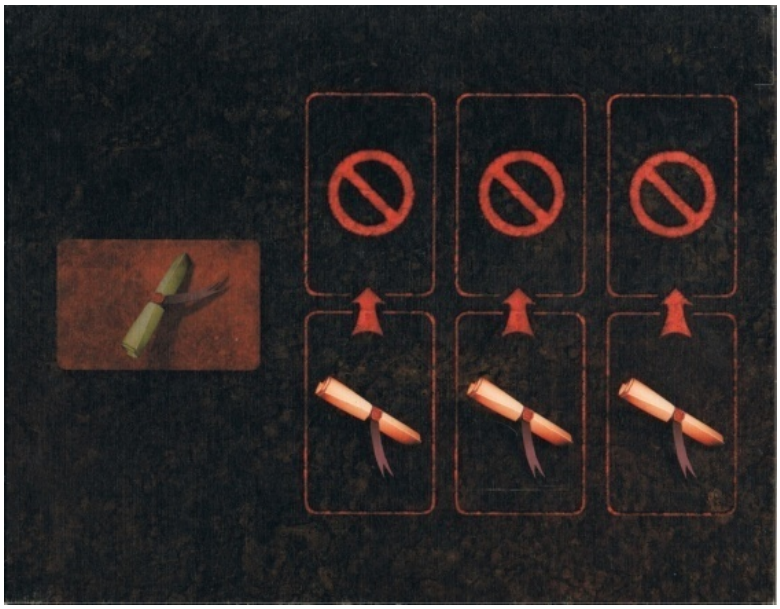
Vzdálené kraje obsahují ty věci, které zatím do hry vůbec nepřišly, anebo věci které se již ve hře neobjeví. Slouží jako určitá forma odkládání.

2.2.4 Hráčské desky

Každý hráč si na začátku hry vybere barvu, za kterou chce hrát a obdrží svou hrací desku. Při převzetí této desky získá hráč také 6 karet příkazů. Při budovací části si vybírá z těchto šesti příkazů 3, tyto vybrané karty vloží na vyznačené místo na desce. Hraní příkazu provede pomocí figurek služebníků, kteří mají také své místo na hráčské desce. Služebníky přidělí na patřičné pole na centrálním plánu. Další nezbytnou součástí je dungeon, místo pro ukládání jídla, zlata a impů. V případě, že tuto hru

Obr. 3: Deska hrajícího hráče

nehraje maximální počet možných hráčů, jsou nehrající desky ovládány zvolenými hráči a pravidla hry jsou mírně odlišná.



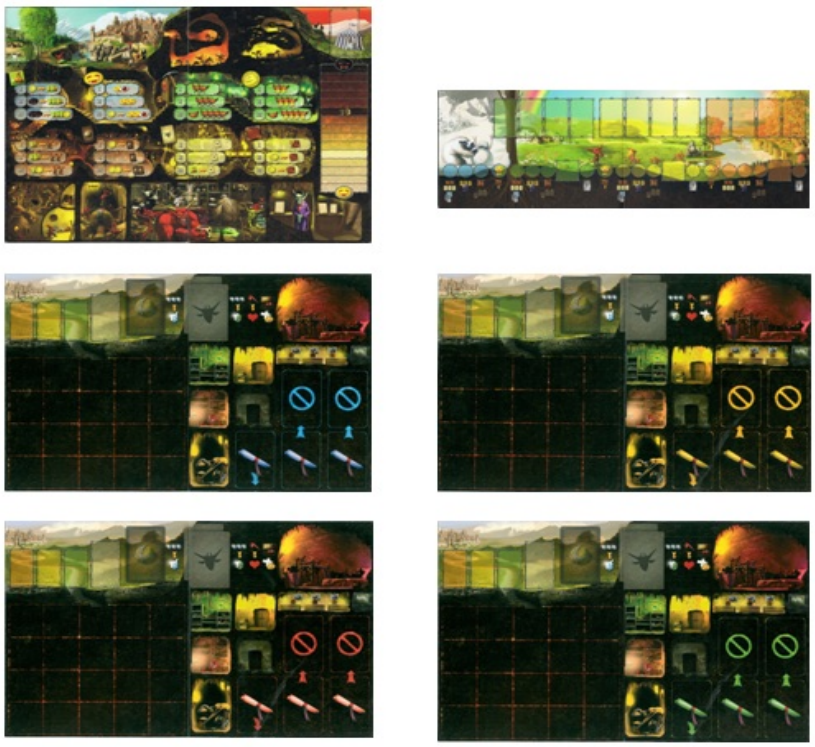
Obr. 4: Deska nehrajícího hráče

3 Návrh řešení

Součástí zadání je i návrh možného řešení, některé věci zde navržené se později při implementaci mohou zdát nevyhovující a budou implementovány jinak. Nicméně si zde popíšeme, jak je možné tento problém řešit. Jelikož hra je vytvořena v jazyku C# v prostředí Microsoft Visual Studia 2010, bylo by dobré využít možnosti rozdělit řešení na více projektů a to na grafické uživatelské rozhraní, kde se bude nacházet vykreslování aktuálních informací o hře, dále pak projekt ve kterém bude samotné řízení hry a hlídání pravidel. Tato možnost je dobrá do budoucna, pokud se stane, že zobrazení hry je zastaralé, nebo z jakéhokoli důvodu by ho bylo třeba vyměnit. V tomto případě se tedy může použít logika hry a vymění se pouze GUI hry. Jako samostatný projekt stojí za to vytvořit i projekt pro síťovou komunikaci, aby veškerá komunikace byla na jednom místě. Rozdělení řešení do více projektu má výhodu pro lepší orientaci v řešení a případnou opravu a jak již bylo řečeno výše, tak i přemostitelnost. Další popis návrhu bych tedy rozdělil na tyto tři části.

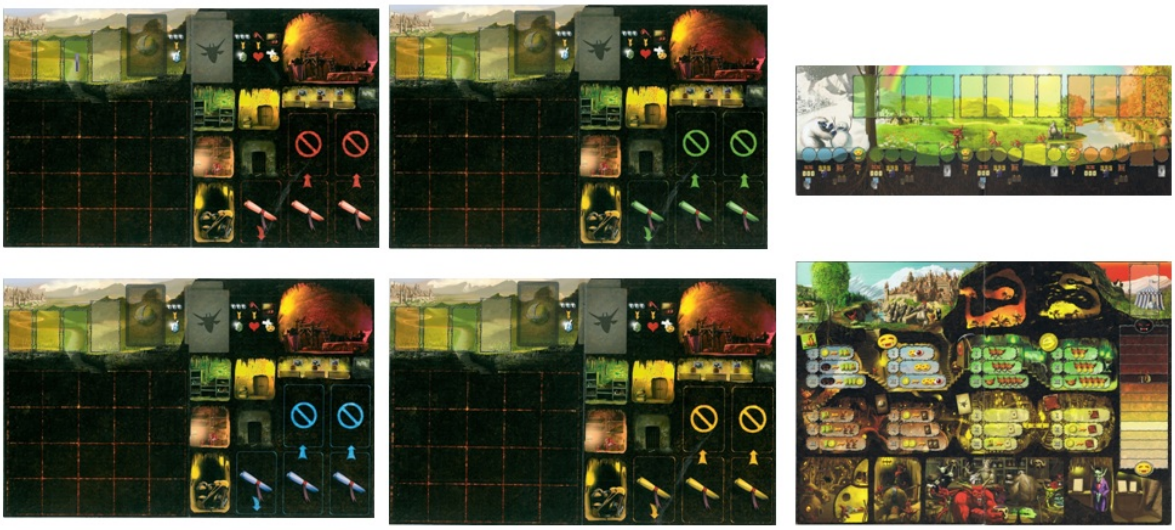
3.1 Návrh grafického uživatelského rozhraní

Vytvoření uspokojivého GUI pro tento program je vcelku problematický. Důvodem je, že na hrací ploše je mnoho hracích plánu k zobrazení. Na ploše bude vždy šest hracích desek, které je potřeba rozložit po obrazovce. Původní nápad, který je na obrázku níže (Obr. 5) jsem nakonec zamítl a to



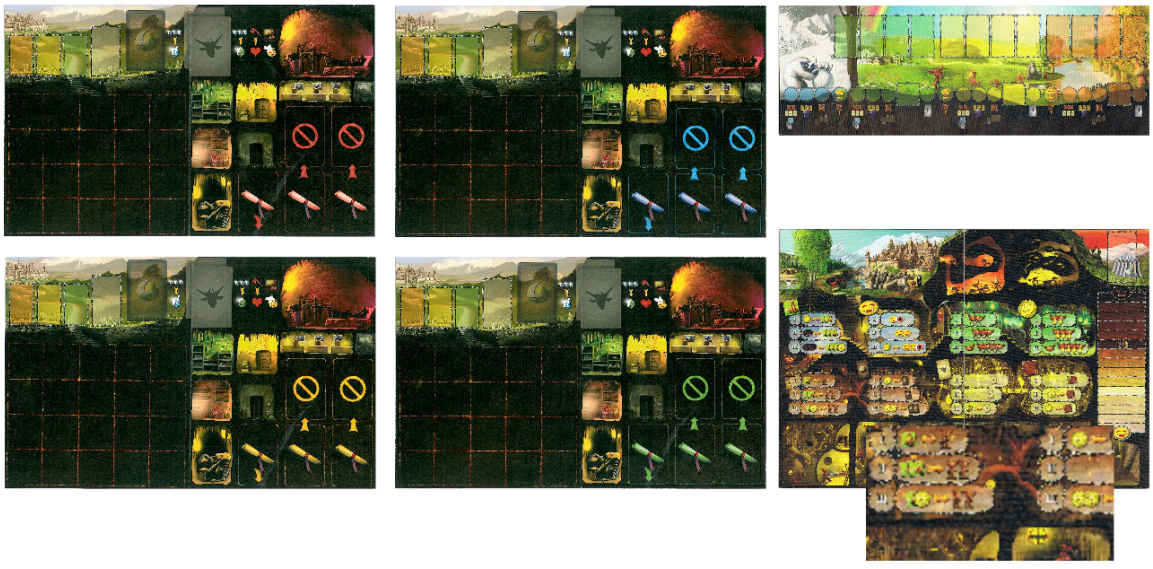
Obr. 5: Vertikální návrh rozložení hracích desek na obrazovce.

z důvodu, že se v dnešní době využívají více širokoúhlé monitory. Výsledný návrh rozložení plánů na hracím poli je uveden na Obr. 6. Při bojové části se časový plán vymění za bojový plán a na začátku nového roku zase opačně.



Obr. 6: Horizontální návrh rozložení hracích desek na obrazovce.

Velké množství hracích plánů vytváří další problém, a to zobrazování detailů na obrazovce. Je nutné, aby každý hráč viděl na obrazovce to, co potřebuje. Jedna z variant jak tento problém vyřešit je vytvoření malého okna vedle kurzoru myši, které při najetí na plán zobrazí detail nacházejíce se kolem kurzoru myši. Toto řešení by ovšem vyžadovalo velkou přesnost hráčů při výběru prvků z hlavní desky. Realizace této varianty je možná pomoci *Canvasu*[2], ve kterém budou všechny plány

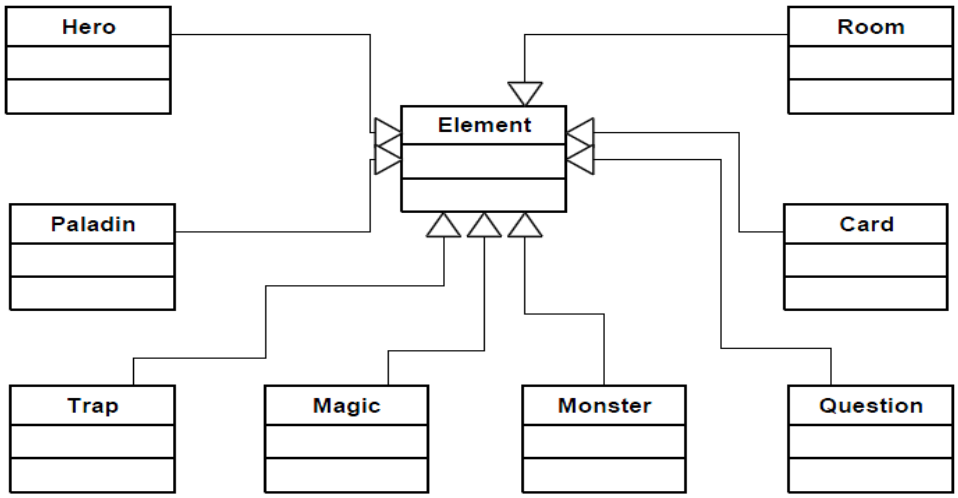


Obr. 7: Ukázka zoomu

a hrací prvky. Při najetí by se z *Canvasu* vyřízl požadovaný kousek plátna a ten by se poté dvakrát zvětšil. Výsledek je zobrazen na Obr. 7.

3.2 Návrh logiky hry

Je potřeba navrhnout správné rozdělení tříd v tomto řešení. Jelikož se jedná o existující hru, tak mnoho tříd bude odpovídat reálné položce v deskové hře. Ve hře je velké množství položek a bylo by dobré je specifikovat. Jistě by měla existovat řídící třída jménem *Game*, která bude mít na starost správný postup ve hře. V rámci lepší přehlednosti je potřeba oddělit třídy hráčů deskami. Herní desky budou tedy vlastnosti hráčů.



Obr. 8: Návrh dědičnosti tříd.

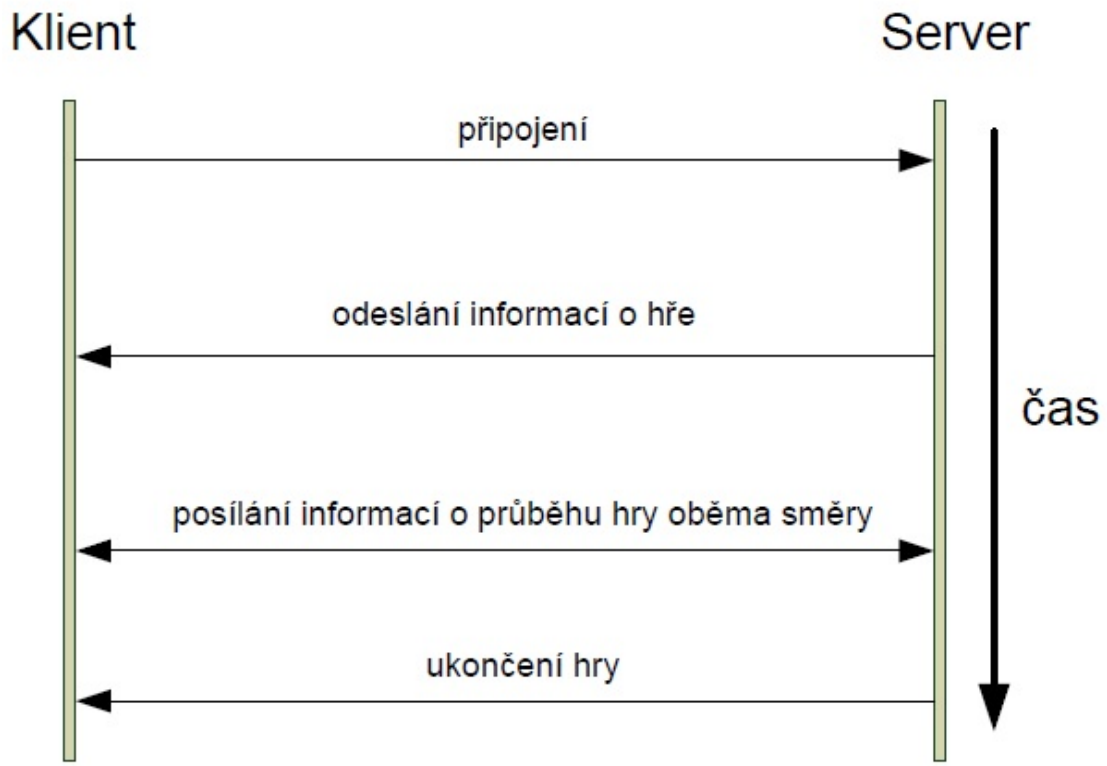
Další věcí, která je důležitá navrhnout, je práce s herními elementy. Nejlepší bude vytvořit jednu počáteční třídu jménem *Element*, která bude obsahovat společné metody a vlastnosti. Od této třídy budou dědit[3] již specifičtější třídy (místnosti, karty příkazů, monstra, hrdinové). Tyto třídy pravděpodobně nebudou konečné a budou od nich dále dědit další prvky. Návrh této dědičnosti je zobrazen na obrázku výše Obr. 8.

3.3 Návrh komunikačního protokolu

Součástí zadání této práce je také vytvoření síťové hry. Proto zde navrhneme komunikační protokol[4], pomoci kterého budou hráči mezi sebou komunikovat. Uvažoval jsem o možnosti vytvoření samostatného serveru někde mimo hráče na který by hráči vyslali požadavek o založení hry a ten by přijímal informace od všech hráčů, dával by je dohromady a informoval by zpětně všechny

hráče o aktuálním stavu hry. Nakonec jsem se ale rozhodl pro variantu, kdy server bude součástí zakládajícího hráče, takže nebude muset být server umístěný někde mimo. Problém nastane v případě, že je zakládající hráč v lokální síti a jeho adresa je do Internetu překládána (NAT). Jiní hráči jsou však mimo tuto síť, musí se tedy poté k němu připojit pomocí sítě VPN, kterou umí vytvořit např. program Hamachi.

Zakládající hráč určí počet hráčů a port na jakém bude jeho server naslouchat. Port společně s IP adresou musí být znám všem ostatním hráčům. Hráči se poté na něj připojí. Server bude nyní čekat na přijetí požadavku o připojení. Jakmile se dočká požadovaného počtu hráčů, odešle všem klientům informace o hře. Poté může nastat vzájemná komunikace mezi serverem a klientem dokud server neukončí spojení. Více Obr. 9.



Obr. 9: Komunikační protokol

4 Implementace hry na PC

4.1 Rozdělení řešení

Tato hra je implementována v jazyce C# v prostředí Visual Studio 2010 .NET Framework 4.0. V řešení je mnoho možnosti, které lze využít. Jedním z nich je rozdělení řešení do více projektů. Toto řešení si dále popíšeme.

Projekt pro zobrazení GUI (grafické uživatelské rozhraní) se nazývá stejnojmenně *gui* a je implementován pomoci WPF. V tomto projektu můžeme vidět veškeré důležité komunikační dialogy s hráčem.

Další důležitý projekt se jmenuje *program*. V tomto projektu lze nalézt veškerou řídící logiku hry včetně veškerých objektů hry (monstra, pasti, kouzla...).

Dle zadání má mít hra možnost hry po síti. Pro veškerou komunikaci (příjem dat, odeslání dat) je zde vytvořen projekt *server.*

Z důvodu, že nemůžeme projekty na sebe cyklicky odkazovat (Projekt program nemůže obsahovat odkaz na projekt *gui.* Pokud již existuje *gui* obsahuje odkaz na projekt program) byl vytvořen čtvrtý projekt pro sdílené datové typy obou projektů jménem *typs*.

V následujícím popisu tvorby této hry budu pokračovat dle uvedených projektů.

4.2 Grafické uživatelské rozhraní

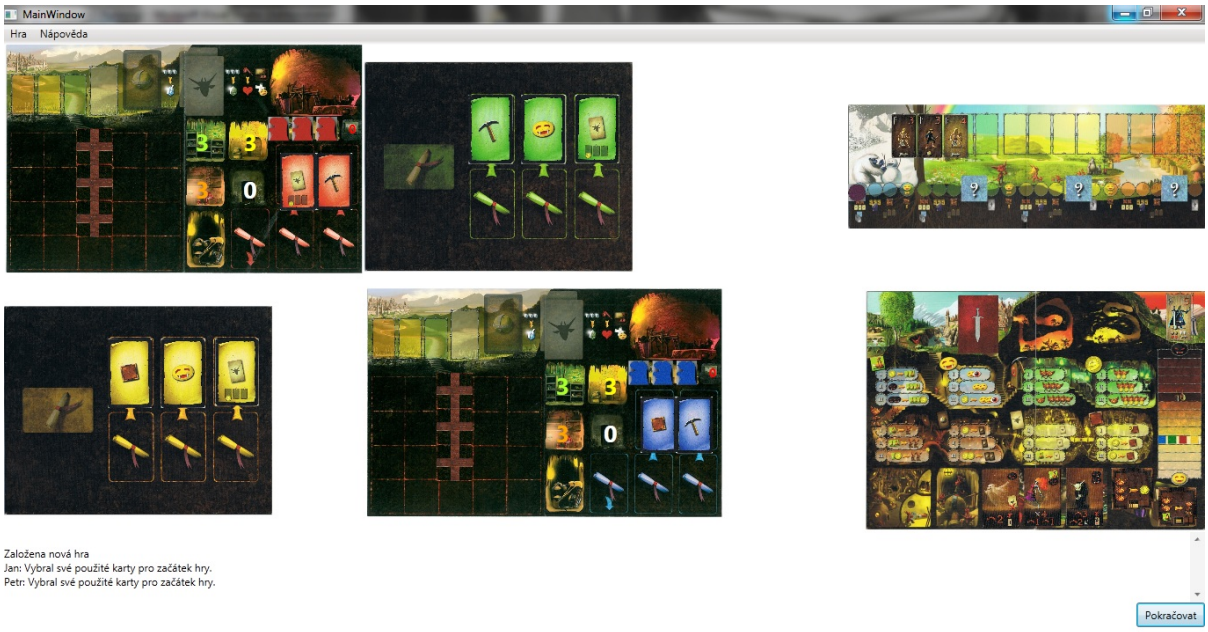
Tato část řešení obsahuje veškeré komunikační dialogy s hráčem/hráči. Nachází se v projektu *gui* a je založena na hlavním okně, které se zobrazí okamžitě po zapnutí hry. Do tohoto okna se po vytvoření nové hry vloží všechny hrací plány. Je zde také mnoho dialogových oken, které slouží k dotazování se uživatele na jeho aktuální tah.

Na Obr. 7 lze vidět návrh hry zoomování. Tato varianta zobrazování detailů bohužel neuspěla, a to z důvodu problému s *Canvasem* . Problém spočíval v občasném neukazování aktuálních informací, v přiblíženém okně však aktuální informace již byla. Po několika konzultacích jsem se rozhodl od tohoto řešení upustit. K zobrazování detailů tedy slouží výše zmíněná dialogová okna.

4.2.1 Hlavní okno

Hlavní okno (*MainWindow*) slouží k vykreslování aktuálních informaci o hře. Při vytvoření nové hry se pokaždé na tomto okně zobrazí všech šest hracích plánů. I při hře v méně než maximálním možném počtu. V dolní části okna se nachází *TextBlock,* který vypisuje aktuální provedené akce

hráčů. *TextBlock* má za úkol odstranit jeden problém, který spočívá ve velkém množství hracích plánů a prvků. Je velmi složité zastihnout veškeré možné detaily ve hře.



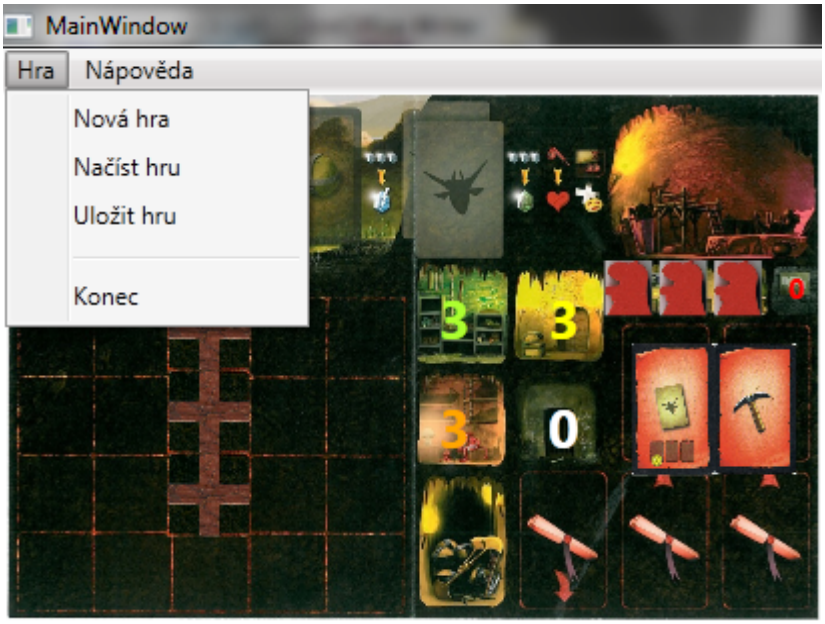
Obr. 10: Ukázka založené hry.

Na obrázku výše uvedeném můžeme vidět hru dvou hrajících hráčů (červený a žlutý). Lze také vidět aktuální nabídku monster a místností. Na časovém plánu můžeme vidět již připravené hrdiny, kteří se přidělí hráčům. Po stisknutí na tlačítko *Pokračovat,* hra spustí *hrací cyklus* (Při síťové hře musí každý hráč stisknout na toto tlačítko). Toto tlačítko bylo do hry přidáno pro možnost ukládání a načítání hry.

Hra v hracím cyklu je plně řízená logikou hry a dovoluje hráči provést jenom aktuálně možné proveditelně věci. Pomocí modálních dialogových oken, které jsou seřazeny správně za sebou. Není tedy možné procházet pomoci *ScrollWiewer* starší dokončené události a také manipulovat s *Menu* v horní části okna. Tento cyklus trvá jedno herní roční období (vyznačeno na časovém plánu) a také mezi útoky hrdinů na podzemí. Celkově je ve hře 16 okamžiků, kdy je možno manipulovat s hlavním oknem. Hra je tímto mnohem plynulejší.

V horní části okna můžeme vidět menu s položkami *Hra* a *Nápověda.* Pod položkou *Nápověda* se skrývá položka *Pravidla hry.* Při výběru této položky se otevře dialogové okno, do kterého se načtou pravidla hry[1]. Po výběru položky *Hra* se zobrazí menu, viz Obr. 11. Z tohoto menu se může hra ukončit, načíst již uložená či uložit aktuální hra. Hra se ukládá v binární podobě, zamezí se tak případnému podvádění. V případě, že byla hra uložena v XML souboru mohli by hráči v uloženém souborů měnit hodnoty (např: množství zlata). V tomto případě vlastnost třídy *MainWindow* jménem

Game se celá binárně serializuje a uloží do souboru. Bylo nutné ručně nastavit, aby se veškeré události (*event*) neserializovaly, protože serializátor chtěl automaticky serializovat třídy. Při načítání síťové hry se hra pomoci dialogu následně zeptá na port, na jakém má hra naslouchat.



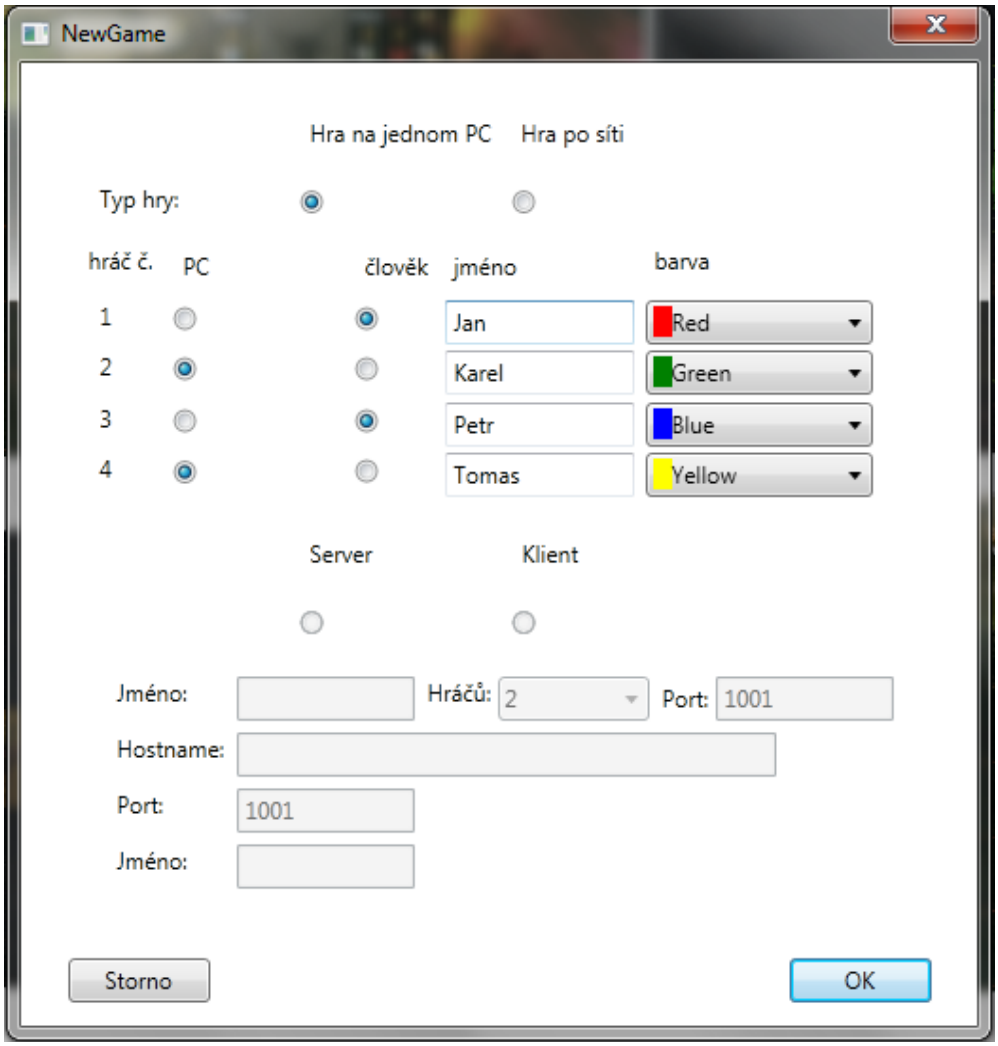
Obr. 11: Ukázka menu v hlavním okně

Po kliknutí na položku *Nová hra* se vytvoří instance třídy *NewGame,* což je dialogové okno pro vytvoření nové hry. Následně se v něm nastaví potřebné parametry pro založení nové hry (počet hráčů, barva plánů, lokální nebo síťová hra...).

4.2.2 Založení nové hry

Jak již bylo řečeno výše, nová hra se vytváří v objektu *NewGame* viz. *Obr. 13,* v tomto dialogu se dá nastavit, kolik hrajících hráčů bude tuto hru hrát, včetně jmen hráčů. Každý hráč si může vybrat, jakou barvu chce, ale při konečném stisku tlačítka *Ok* se nesmí stát, aby existovaly stejné barvy u hráčů. Každý si může vybrat jméno podle své libosti. Pokud tuto možnost nezvolí, bude mu přiděleno jméno typu: „hráč č.“ s číslem na kterém se hráč nachází. Hráč si v tomto okně taky vybírá, zda se bude jednat o lokální hru, či síťovou. Pokud se jedná o lokální, pak nastavuje parametry uvedené výše.

Nastavení hry pro síťovou hru se děje po stisku na *RadioButton* jménem *Hra po síti*. Následně si hráč vybere, zda se jedna o hráče, který hru zakládá, čili se jedná o s*erver*. Při výběru této možnosti zakládající hráč musí určit, na jakém portu bude server naslouchat, počet hrajících hráčů a jméno sebe sama jako hráče. Jméno hráče je také libovolné. Pokud se ale stane, že server zjistí, že se ve hře vyskytují stejná jména, hra přidá každému takovému hráči ke jménu číslo jeho pořadí výskytu tohoto jména. Je to z toho důvodu, že se při nahrávání síťové hry, hráči zpětně určují podle jmen hráčů. Barvu plánů si hráči při síťové hře nemůžou sami vybrat je jim automaticky přiřazena podle jejich pozice ve hře a pozice je určena podle pořadí přijetí jejich požadavku k připojení serveru. Pokud si hráč vybere hrát, jako *Klient* musí uvést kromě svého jména i IP adresu, na kterém se nachází server a port na kterém hra naslouchá. Je tedy nutné, aby se předem hráči na těchto parametrech domluvili.



Obr. 13: Založení nové hry.

Se jménem platí stejná pravidla, jako při založení síťové hry. Barvu a pozici hráče určí server podle jeho pořadí při připojení na serveru. Při připojování ke hře, která byla na serveru nahrána ze souboru.

Se připojuje ke serveru stejným způsobem, je ale třeba si dávat pozor jestli naše jméno odpovídá, pokud neodpovídá server hráče odmítne.

Veškeré nastavení tohoto okna se automaticky, ukládá do dokumentů aktuálního uživatele operačního systému. Neukládá se přímo k programu, protože se program může nacházet v místech, kde není povolený zápis. Pokud je hra spuštěna poprvé, soubor se v dokumentech uživatele vytvoří a zobrazí se výchozí nastavené hodnoty. Při znovu zapnutí programu se zobrazí posledně vyplněné parametry. Je to praktické z důvodu šetření času při vyplňování neustálé stejných hodnot.

Po stisknutí na tlačítko *OK* se provede kontrola zda jsou hodnoty správně vyplněny a to především zda jsou zvoleni minimálně dva hrající hráči, dále kontrola zda nějací hráči nemají nastavené stejné barvy (je to z důvodu toho, že pokud by byly stejně zbarvení úředníci na hlavní desce, pak by nešlo rozpoznat, o jakého hráče se jedná). Při stisku tlačítka *Storno* se žádné změny neprojeví a ani se neuloží do XML dokumentu s nastavením.

Po potvrzení a úspěšném vytvoření hry se vytvoří v hlavním okně odpovídající plány. K tomuto slouží v projektu *gui* složka *imageWork,* ve které jsou statické třídy potřebné pro vytvoření odpovídajících plánu (*MakeDesk*) na hlavní okno. Následně již specifikované statické třídy vytvořily odpovídající mřížku (*MakeMain, MakePlay, MakeNoPlay...*), do kterých se na specifická místa umísťují instance třídy *Item* nebo *Description*. Jedna z vlastností třídy *Item* je položka typu *Image,* která slouží k zobrazení položky ze hry (monstrum, past, kouzlo...). Ve třídě *Description* se nachází vlastnost typu *Label,* která zobrazí aktuální hodnoty (např. zlata). Objekty typu *Description* se nacházejí pouze na plánu hrajícího hráče. Při vytváření mřížek desek byla nejdříve potřeba vytvoření dané mřížky v XAML v okně *MainWindow* a následné převedení pro dynamické vytváření v programu.

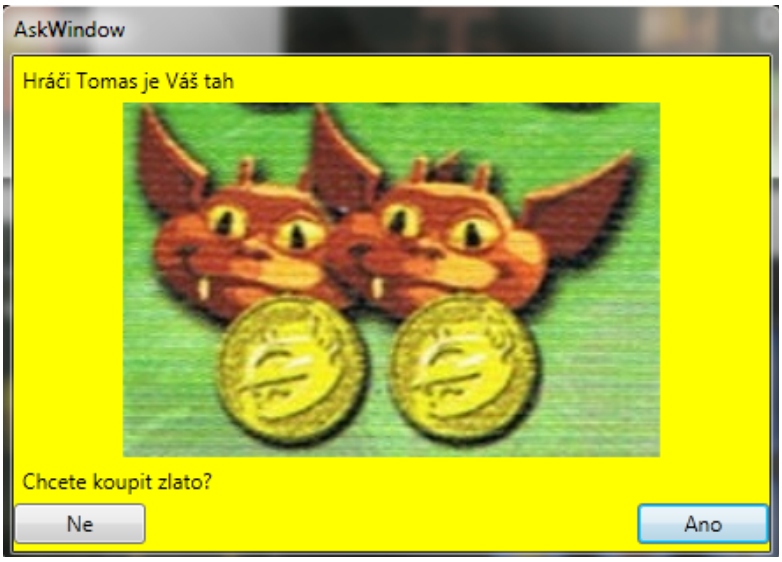
Jakmile se vytvoří odpovídající panel a mřížka, podle nastavení hráče, tak se následně propojí vytvořená hra s hlavní oknem a jejími deskami. Toto se provede pomocí událostí (*events*), tyto události jsou součástí tříd v herní logice. V této částí se pomocí tříd v souboru *imageWork/events* navážou metody k událostem. Dále se budou provádět také při každém volání události. Provázání události není součásti ukládání, proto se musí při nahrávání hry znovu provázat.

Všechny tyto události mají za následek vytvoření dialogového okna, ve kterém se hráč rozhodne podle své libosti. Většina těchto oken je určena pro rozhodnutí aktuálního hráče. Z tohoto důvodu byla potřeba nějakým způsobem specifikovat, o jakého hráče se jedná (tento problém nastává pouze při lokální hře). Při zobrazení těchto oken se pozadí dialogového okna změní na barvu aktuálního hráče a vlevo nahoře se vypíše popisek „Hráči (jméno) je Váš tah“. Rozhodovací dialogová okna se nacházejí v souboru *askWindow.* I přesto, že jsem se snažil dělat okna multifunkční (využít jedno okno na více volání události) jejich výsledný počet je třicet. Jedním z důvodů, proč jich je tolik je fakt, že řeší problém se zobrazováním detailů. Veškeré rozhodování

hráče se odehrává v dostatečně velké velikosti a nemusí se tedy řešit malé detaily. Pro uvedení příkladu dialogových oken popíši vždy jedno okno za každou kategorii.

4.2.3 Dotazující se okno

Toto okno je ve třídě *AskWindow* (Obr. 14). Slouží k dotazování hráče, jestli chce provést nějakou akci. Konkrétně to je v případech, kdy se logika hry ptá, jestli chce využít úředníka, který stojí někde v nákupním poli.(př: „Chcete koupit jídlo?“). Hráč má možnost buď souhlasit, nebo odebrat svého úředníka z hlavní desky a nevyužít jeho postavení v obchodě. Toto rozhodnutí je důležité z důvodu, kdy chceme protihráče jenom blokovat, ale sami tuto možnost nepotřebujeme využít, nebo jsme se dostali do chtěného obchodu, ale ne za podmínek, v jaké jsme doufali.



Obr. 14: Příklad dotazujícího se okna

4.2.4 Okna s kombo boxem

Tato okna mají v sobě *ComboBox* pro výběr jednoho prvků, je to využito třeba při výběru monster pro obranu podzemí, výběr pastí. Aktivní hráč si vybere z *ComboBoxu* prvek a ten se následně zobrazí pod boxem. Je to použito při výběru prvků, kdy nemůžeme vědět jistě, kolik daných prvků bude k výběru. U monster jich může být 0-8, pastí může bát daleko více.

Na níže uvedeném obrázku (Obr. 15) je uvedený příklad pro placení znova cenu všech monster. Tady je výčet všech oken, které fungují na tomto principu, svoji funkčností se, ale liší: *AuraOfFear, DestroydTrap, DetectionTreasuer, EatMonster, ChooseDefenceTrap, ChooseChoust, ChooseMonster, Metamorphosis, PayMonsters.* Tyto okna využívají objekty třídy *ComboData,* která používám pro snadnější vložení dat do *ComboBoxu.*

Obr. 15: Okno s Kombo boxem.

4.2.5 Okno podzemí

Tato okna se využijí v případě, kdy hráč chce změnit něco ve svém podzemním komplexu. Může to být kopání nových chodbiček, těžba zlata atd.. Tato dialogová okna načtou nejdříve podzemí z původního zdroje, který se zobrazuje zároveň na hlavním okně. Veškeré provedené změny se okamžitě projeví hlavním okně, i když práce na podzemí ještě nebyla dokončena.

V příkladu podzemních oken (Obr. 16) jsem vybral okno, kdy se vkládá do podzemí nová koupená místnost. Vybraná místnost je zobrazena v pravé horní části tohoto okna. Obrázek zvolené místnosti jsem přidal z toho důvodu, abychom věděli kde danou místnost vložit (vyznačené žluté kostičky v dolní části místnosti). Hra nám nedovolí vložit místnost na nepovolená místa. K tomu slouží výpočetní algoritmus, který je součásti logiky hry. Zvolenou pozici můžeme po kliknutí na aktuální pozici místnosti zrušit a zvolit jinou. Konečný stav se uloží při stisknutí na tlačítko OK.

Do této kategorie spadají tyto třídy: *DigGold, ChooseDefenceRoom, InsertCorridor, InsertRoom, PayRooms, SetImps.*

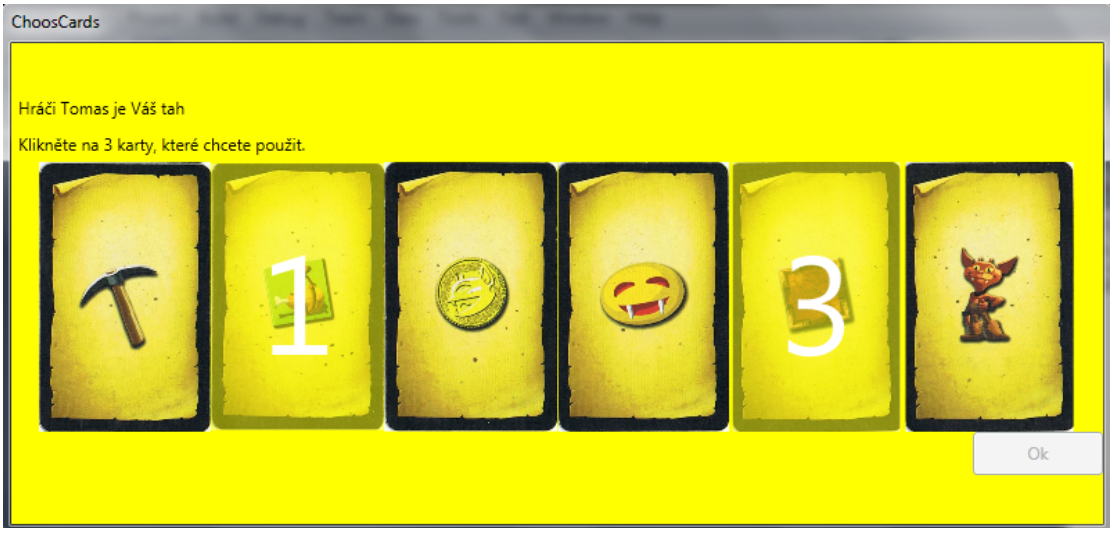
Obr. 16: Ukázka okna pro práci s podzemím

4.2.6 Výběr prvků

Další okno, které bych zde chtěl popsat je pro výběr karet, monster, pasti. Používají se v případě, kdy předem přibližně, víme kolik prvku se v okně zobrazí. Výběr se provádí pomoci kliknutí na chtěný prvek a následné potvrzení tlačítkem *OK*. Tento výběr si může hráč rozmyslet do bodu, než stiskne tlačítko *OK*, zrušení výběru se provede opětovným stiskem na vybraný prvek a následně vybrat nový.

Vybral jsem zde pro ukázku výběr hracích karet před začátkem, každého hracího obdobu. Je to jedno ze dvou speciálních oken tohoto druhu pro výběr více karet a to 3. Hráči se na vybrané kartě ukazuje číslo, v jakém je karta pořadí. Pokud se hráč rozhodne vybrat jinou kartu na jakoukoli pozici, stačí vybranou kartu odkliknout a vybrat novou. Po definitivním výběru hráč musí potvrdit svou volbu na tlačítku *OK*.

Okna s tou funkcionalitou se jmenují: *ChooseCards, ChooseItem, ChooseRoom, ChooseTrap, ChooseUsedCard, StartChooseCards*. Okno *StartChooseCards*se zobrazí okamžitě po vytvoření nové hry, slouží pro výběr nehrajících karet, před začátkem hry.



Obr. 17: Ukázka výběru karet.

4.2.7 Výběr Útoků

Obr. 18: Výběr útoku monstra.

Výběr útoku je okno, které se zobrazí, pokud hráč vybere pro obranu svého podzemí monstrum, které má jako možnost k výběru dva různé útoky. Hráč pouze klikne na jedno ze svých tlačítek. *Primární*

či *Sekundarní* útok. V projektu *gui* se tato třída jmenuje *ChooseAtack.* Toto okno je stejné pro všechna monstra tohoto druhu, mění se pouze obrázek monstra a reakce na výběr.

4.2.8 Výběr zraněného hrdiny

Obr. 19: Ukázka výběru zraněného hrdiny.

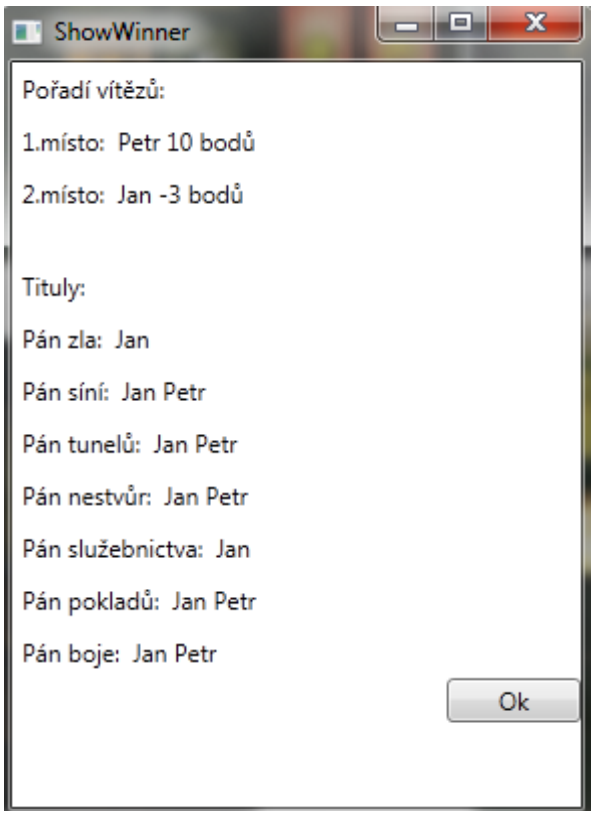
Toto okno je také jediného svého druhu a nachází se ve třídě *ChooseHearHero.* Používá se u monster nebo pastí, kde potřebujeme určit, který z hrdinů byl tímto útokem zraněn. Většinou se po kliknutí na nějakého hrdinu, okno zavře (pokud byly splněny podmínky útoku). V případě čarodějnice se čeká na druhý útok.

4.2.9 Ukázání prvku a ukázání vítěze

Tato dvě okna slouží pouze k zobrazování informací. Může to být v případě aktuálního kouzla, získané pastí, zobrazení výsledku hry... .

Pro ukázku (Obr. 21) jsem zde vybral okno pro zobrazení vítěze hry. V původní deskové verzi se vítěz hry zdlouhavě počítal podle zvítězených titulů a získaných bodu, v případě PC verze to hra vypočítá sama a zobrazí i získané tituly za každého hráče. Samozřejmě nezapomene odečíst trestné body.

Obr. 20: Příklad zobrazení kouzla.



Obr. 21: Ukázka ukončení hry.

4.3 Logická část řešení

Logická část hry, jak již bylo výše uvedeno, se nachází v projektu *program.* Veškeré následující třídy spadají pod ni. V tomto projektu se nacházejí čtyři základní soubory, které budu dále podrobněji popisovat. Prvním souborem je složka *element,* v této složce se nacházejí veškeré prvky hry. Jedná se o monstra, hrdiny, pasti,.... Další soubor se jmenuje *boards,* a zde jsou uloženy veškeré třídy herních desek.V těchto deskách lze najít události, které se volají pro vykreslení *gui*. Souvisejícím souborem je *players,* ve kterém jsou všechny typy hráčů (hráči mají ve svých vlastnostech také instance ze souboru *boards)*. Poslední soubor, do kterého se v této části dostaneme, se jmenuje *Game.* V této třídě je uloženo samotné řízení hry a také se zde nachází jediný soubor, a to *Game.* Tento projekt po vytvoření nové hry přebírá celkové řízení hry.

Vytvoření hry v tomto kontextu znamená, vytvoření odpovídajících hráčů (případně vytvoření spojení, nebo připojení hráče k serveru). Tito hráči se používají jako parametr pro vytvoření instance třídy *Game*. Po vytvoření objektu hra, se provážou události hráčů a herních desek s projektem *gui* (popsáno výše). Následuje náhodné vybrání použitých karet nehrajících hráčů. Poté si každý hrající

hráč vybere jednu kartu ze tří náhodně vybraných karet, kterou bude moci používat v prvním období. Hra bude následně čekat na stisk tlačítka *Pokračovat*.

4.3.1 Složka s herními elementy

Tato složka obsahuje sedm podsložek herních elementů. Tyto elementy jsem naskenoval z původního deskového provedení a vyřezal do konečného provedení. Grafické zobrazení těchto elementů je k nalezení v projektu *gui* souboru *Images*. Dále se budeme zabývat pouze jejich funkcionalitou, protože o samotné vykreslení se zabývají *gui.* Budeme se tedy dále řídit rozdělením podle výše uvedených sedmi složek.

Rozdělení do sedmi složek odpovídá návrhu, ale ukázalo se zbytečným mít jednu společnou třídu *Element.* Prvky ve hře nemají mnoho společného, tudíž vytvoření bázové třídy *Element* se zdá zbytečné.

Karty příkazů

Karty příkazů se nachází v souboru *cards,* jejím jediným obsahem je třída nazvána *ItemsCard.* Obsahuje dva typy vlastností karet (kopání chodbiček, monstra, pasti...) a pravdivostní proměnnou *Used,* která určuje zda byla karta použita. To znamená, že se na hracím plánu otočí lícem nahoru. Každý hráč má těchto karet celkem osm. Tyto karty se vytvoří při založení hry a během hry se přesouvají mezi balíčkem použitým, používaným a nepoužívaným.

Hrdinové a paladinové

Následující elementy bylo možno, jako poslední z elementů, sjednotit do tří tříd. První třída jménem *Hero* obsahuje popis hrdinů a seřazení podle šesti vlastností (tyto informace jsou součásti parametru konstruktoru). Typy hrdinů jsou následující: válečník, zloděj, kněz a kouzelník. Informace o jeho obtížnosti je uložena hned za typem hrdiny. Dále je specifikována míra jeho speciální vlastnosti (pokud se jedná o válečníka je vždy 0). Další dvě položky si jsou velmi podobné, první z nich nám udává maximální možný počet životů hrdiny a ta druhá ukazuje aktuální počet životů (na začátku hry se obě hodnoty rovnají). Poslední dvě hodnoty slouží k rozpoznání grafického znázornění v *gui.* Jedná se o herní rok, ve kterém se daný hrdina vyskytuje a poslední hodnotou je pevné pořadí, které jsem při vytváření specifikoval.

Třída jménem *Paladin* je podobná jako třída *Hero,* slouží ale k vytváření paladinů. Ve hře jsou pouze 2. U paladinů se vyskytují jiné vlastnosti než u hrdinů. První tři jsou hodnoty jeho každé speciální schopnosti. Obsahuje také vlastnosti na maximální počet životů a aktuální počet životů. Vlastnosti pro určení roku a pořadí ovšem neobsahuje, *gui* pozná jaký obrázek použít pro paladina podle jeho maximálního počtu životů.

Poslední třída v tomto souboru se nazývá *Heros* a má veřejné vlastnosti, ve kterých jsou obsaženy balíčky hrdinů pro každý rok. Uspořádání hrdinů je vždy náhodné. Jsou zde také veřejné dvě celočíselné vlastnosti, které nám určují pozice v těchto balíčcích. Náhodný výběr pořadí hrdinů probíhá nejdříve vytvořením listu seřazených hrdinů pro každý rok. Z tohoto listu poté náhodně vytahuji hrdiny a vkládám je do veřejných listů. Tento způsob „kopání“ balíčků používám v každém náhodném baličku ve hře (např: monstra, pasti). V této třídě jsou také oba paladinové.

Kouzla

Kouzla jsou vytvářena pomocí dědičnosti abstraktní třídy. Třída *Magic* pojmenovává a určuje, jak a s jakými proměnný se bude metoda volat a také určuje její návratový typ. V této třídě se deklaruje šest různých metod *useMagic,* které slouží k použití aktuálního kouzla na hráče. Argumentem této metody je tedy postižený hráč. V případě provedení kouzla na více hráčů je třeba zavolat tuto metodu na každého hráče zvlášť. Další metoda v pořadí, má název *used.* Tato metoda se používá ke zjištění, zda již bylo kouzlo ukázáno. Pokud ano, zobrazí se lícem nahoru. Toto zjištění je určeno pro *gui*. Metoda *show* nastavuje vnitřní proměnou, díky které se karta ukáže (otočí se lícem nahoru). Dvě následující metody rozlišují rok, pro jaký bylo kouzlo určeno a také typ kouzla, zda se jedná o rychlé či pomalé kouzlo. Poslední metoda má název *getBlood* a vrací počet životů, které tento útok hrdinům strhne.

Jelikož se jedná o abstraktní třídu, všechna kouzla od třídy *Magic* musejí tyto metody implementovat. Každé kouzlo má tedy specifickou implementaci těchto metod. Kouzla jsou rozdělena do dvou složek, dle roku pro které jsou určena. U některých kouzel mají hráči na výběr z více možností. Pro jakou z možností se rozhodnou je pouze na nich (např: *WordReconciled* dává hráčům na výběr, zda stáhnou své monstrum z boje, nebo se posunou o dva pole směrem ke zlu). V takových případech se vyvolá událost pro výběr reakce hráče.

Další třída, která spadá pod tuto oblast je *Magics.* Tato třída vytváří dva náhodné baličky sestavené ze čtyř kouzel. Jeden balíček je určen právě pro jeden rok. Způsob jakým se baličky na začátku hry vytvářejí je stejný jako u hrdinů a paladinů.

Monstra

Ve složce *monsters* najdeme veškeré monstra, která se během hry objevují. Každé monstrum má ve hře specifické vlastnosti. Vytvoření jedné třídy (např. monstrum) by šlo pouze v případě, že by se jednalo pouze o cenu monster. Tato vytvořená třída by se poté realizovala obdobně jako u hrdinů. Hlavním důvodem proč má každé monstrum svou vlastní třídu je různorodost útoků. Každé monstrum má svůj specifický útok, ale některé prvky útoků jsou stejné (např. útok na prvního, útok na kohokoli a útok na všechny).

Veškerá monstra mají jednu otcovskou abstraktní třídu, která má vlastností a metody, jež monstra dědí. Mají ale také metody, které si musí monstra samy implementovat, jedná se například o výše uvedený útok. Monstrum obsahuje odkaz na svého vlastníka. Tento odkaz je potřebný při placení monster a nastavování útoků monster. Jsou zde také vlastnosti, které obsahují informace o obraně podzemí (zda bylo použito nějaké kouzlo atd.) a určení, zda bylo monstrum zaplaceno či nikoliv. Tato vlastnost se využívá ve chvíli, kdy se ve hře objeví událost, která dává za úkol hráčům zaplatit znovu cenu všech svých monster. Vlastnost je také využitelná při kouzlu. Další ze zde přítomných informací nám říká, zda bylo monstrum tento rok použito k obraně hráčova podzemí. Poslední vlastnost určuje, zda došlo pomocí kouzla k přeměně monstra na ovečku.

Metoda, která byla vytvořena již v otcovské abstraktní třídě, je použita také tady. Jedná se o metodu, která je schopna odstranit monstrum (např. při nezplacení ceny monstra). Potom jsou zde použity metody pro navracení informace o zaplacení monstra, nastavení informace o zaplacení či nezaplacení a nastavení majitele monstra (používá se např. při nahrávání síťové hry). Je zde také virtuální metoda, která umožňuje, že monstra mohou použit tuto implementaci nebo vytvořit vlastní. Tato metoda slouží k posunutí ke zlu v případě, že nemáme na zaplacení monstra.

Stejné jako u kouzel jsem k implementaci ve třídě *monster* použil abstraktní metody ve třídě *monster*. Těmito metodami se v tomto odstavci budeme zabývat. První z nich je *buyMonster*, jak už název napovídá, jedná se o metodu, která bude volána při pokusu o koupi monstra. Metoda může skončit neúspěšně, pokud hráč, který si chce monstrum koupit, nemá dostatek potřebných surovin. Následující metodou je *payMonster,* jejímž úkolem je znovu zaplatit cenu monster, pokud hráč nemá dostatek prostředků pro znovu zaplacení, o monstrum přichází a metoda tedy vrací neúspěch. Jejím parametrem již není hráč, protože je uložen ve vlastnostech, ale pravdivostní proměnná, díky níž se nastavuje, zda se hráč posune při nezaplacení svého monstra směrem ke zlu. Samozřejmě zde nemůže chybět metoda pro útok monstra, jejichž parametrem je balíček příznaku o aktuální obraně. Jsou zde také primární a sekundární útok, u některých monster jsou tyto metody prázdné, protože volbu sekundárního útoku nemají.

Pro vytváření náhodných balíčků monster na každý rok hry je zde vytvořena třída *Monsters,* v její implementaci není nic, co by nebylo vysvětleno při vytváření balíčků jiných elementů.

Herní události

V pravidlech hry je pro lepší pochopení hry zařazena nejdříve neúplná hra, která nepoužívá karty událostí a některé další složité prvky hry. V mé PC verzi jsem karty událostí také nevytvořil. Toto je ovšem jediná neimplementovaná věc v PC verzi, oproti úplné deskové verzi. Zachoval jsem ovšem kompatibilitu s neúplnou deskovou verzi. Některým hráčům to jistě zjednoduší pochopení a samotné hraní hry.

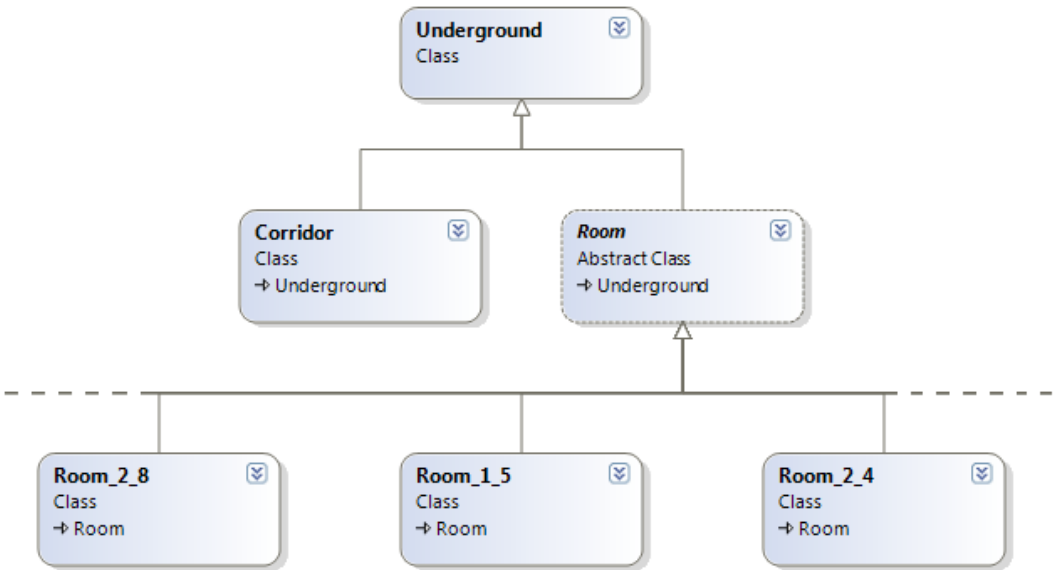
Ve složce *questions* je k nalezení rozhraní pro jakoukoli herní událost ve hře. Každá událost musí tedy obsahovat tři metody. *UseQuestion*, která má jediný parametr, a to *list* hráčů sloužící k provedení události pro všechny hrající hráče. Další metoda se jmenuje *used* a v její kompetenci je navrácení pravdivostní hodnoty, pokud již daná událost byla na obrazovce zobrazena lícem vzhůru. Tato metoda je využívána v *gui.* Poslední metoda je *show.* Tato metoda slouží pro nastavení vnitřních vlastnosti pro otočení herní události lícem vzhůru.

Jsou zde vytvořeny tři herní události. Jedna z nich neprovede žádné akce (v rámci budoucího rozšířeni, by zde šly dodělat karty událostí). Další herní události slouží k zaplacení daně za chodby, včetně místnosti, v podzemí a také znovu zaplaceni ceny monster. Obě tyto události využívají řídicí události pro zobrazení speciálních oken v *gui.* Skrze tato okna provede hráč výběr, jimž bude dále pokračovat.

Je zde také třída *Questions* sloužící k náhodnému rozložení třech události po časové desce. I zde je náhodné rozložení herních událostí vytvořeno stejně jako např. u hrdinů. Tento postup je tedy popsán již výše.

Místnosti a chodbičky

Soubor *rooms* obsahuje zatím nejsložitější dědičnost v řešení. Na obrázku (Obr. 22) můžeme vidět jeho realizaci v Class diagramu. Hlavní třída *Underground* je otcem jakéhokoli místa v podzemí, sama také vyznačuje prázdné místo v podzemí. Obsahuje také informaci o majiteli určitého kousku podzemí. Podává také informaci o stavu zničení podzemí dále o počtu impů v něm a o dvourozměrném poli *listu,* který ukazuje kde je možno postavit tento kousek podzemí.



Obr. 22: Hierarchie dědičnosti místnosti a chodbiček.

Je zde také použito několik metod, první z nich nastavuje majitele podzemí. Další slouží ke zjištění, zda je chodba či místnost dobyta a také k nastavení dané chodby, nebo místnosti na dobytou. Je možno zjistit kolik impů se v této položce nachází. Virtuální metoda *Bay* bere jeden parametr a to hráče, ktery si položku kupuje. Tato metoda má za úkol zjistit, kde se může položka vložit, a kde ne. Využívá se při vytváření chodbiček. Druha důležitá virtuální metoda se jmenuje *GetImps,* která se využívá pro navrácení impů zpět z aktivní položky.

Dědící položka *Corridor* představuje chodbičky v podzemí. Obsahuje čtyři důležité vlastnosti, a to: *DigMasterImp, GoldMasterImp, DigImp* a *GildImp.* Tyto parametry jsou soukromé pro tuto třídu a nastavují se přes metody. Tyto vlastnosti slouží k umístění impa nebo více impů na práci. Ve hře je povoleno na nově postavené chodbičce ihned těžit zlato. Při těžbě zlata, či kopání chodbiček je někdy nutné umístit do podzemí mistra, který dohlíží na práci. V deskové podobě mají tito mistři předepsané místo, musí stát ve vchodu do podzemí. V PC verzi je tento problém vyřešen umístěním dělníka i mistra na první vybranou chodbičku. Může se tedy stát, že na jedné chodbičce budou stát až 4 impi. Kromě metod, které slouží k práci s vlastnostmi impů jsou zde i jiné metody. Metoda pro přidání chodbičky do podzemí, která přijímá dva parametry pro přesné určení pozice v podzemí. Dále je tady přepsána metoda pro získání impů zpět z chodbičky. Je potřeba nastavit všechny výše uvedené vlastnosti na neaktivní.

Nyní se dostáváme k abstraktní třídě *Room.* Tato třída obsahuje statickou vlastnost o roce hry a statickou metodu při nastavení hry pro druhý rok. Dále je také součástí této třídy možnost využití místnosti v druhém roce, a to ve dvojnásobném množství. Je zde také implementována metoda pro vložení položky, čili místnosti do podzemí, která bere dva parametry určující přesnou pozici v podzemí. Také je zde přepsána metoda pro nakupování místnosti, která má jiná pravidla ohledně možných míst pro vložení místnosti do podzemí. Předepisuje třídám metodu *useRoom* kvůli využití jejich speciálních vlastnosti při umístěni impů.

Každá místnost ať je určena pro jakýkoli rok má specifickou implementaci metody *useRoom,* což je činí specifickými a zaslouží si každá svou implementaci. Dvě z těchto tříd *Room_1_2* (získáni pasti) a *Room_1_5* (kopáni chodbičky) jsou výjimečné tím, že vyvolávají události, které potřebuji reakci hráče. Vytvoření dvou baličku pro hru se najde ve třídě *Rooms,* která je výjimečná pouze voláním statické metody (setSecondYear) ve třídě *Room* pro nastavení druhého roku.

V složce *rooms* se nachází ještě jedna třída, jež je pojmenována *Undergrounds.* Tato třída je součásti každé desky hrajícího hráče, obsahuje totiž dvourozměrné pole reprezentující podzemí hráče. Jsou zde funkce pro práci s celým podzemím (např. volání na každý prvek podzemí, aby vrátil impy). Nachází se zde také dle mého názoru nejtěžší algoritmus[5] tohoto řešení. A to vyhledávání nejbližší možné chodbičky pro boj. Na začátku každé obranné části se v této třídě zavolá metoda *GetDefenceRoom,* která vrací zvolenou část podzemí pro dobývání. Pokud je více stejně vzdálených

prvků podzemí od vchodu, vyvolá se událost pro výběr od hráče. V první řadě se po iniciaci zavolá metoda *lookToThisCorridor* (pozice nejblíže ke vchodu), ta se podívám zda je toto místo nedobyto. Pokud ano, vrátí tuto pozici. Naopak pokud je dobyto podívá se pomoci rekurzivního volání do dalších třech směru, a zároveň zkontroluje, zda se do této části podzemí již nepodíval. Tato kontrola je z důvodu, aby nedošlo k zacyklení. Tento princip funguje pro celé podzemí. V případě, že se nenajde žádné další místo možné k boji a následnému možnému dobytí dojde ke ztrátě (pokud to je možné) zajatého hrdiny.

Pasti

Implementace pastí zase nepřináší nic nového, každá past potřebuje svou vlastní metodu pro její samotné použití. Dvě pasti: *AntimagneticArrow* a *PoisonArrow* mají události, které vyvolají nutné rozhodnutí hráče.

Abstraktní třída *Trap* má v sobě jedinou abstraktní metodu, a to *useTrap.* Tato metoda bere jeden parametr, a to hráče, jemuž past patří. Všechny pasti od této třídy dědí. Vytváření balíčku, ve kterém je každá past třikrát se děje ve třídě T*raps.*

4.3.2 Složka s deskami hry

Než se ponořím, do popisu všech desek rád bych na tomto místě uvedl, jakým způsobem se volají události, aby se aktuální věci vykreslily na obrazovku. Pokud se jedná o prvkem uložený na hrací desce, volá se dvěma způsoby. Tou nejjednodušší je pokud se jedná o hodnotové typy (*int, double...*) u těchto vlastností se při volání *Set* automaticky vyvolá událost. Horší to je s referenčními datovými typy, kdy součásti *Set* je také okamžité volání události, jenomže při zvnitřní změně objektu, nebo seznamu objektu, nedojde k vyvolání události. V takových případech vyvolávám událost, tak že přepíšu aktuální referenci sama sebou.

V souboru *boards* se nachází pět desek a jeden soubor s obchody nacházející se v deskové podobě na hlavní desce. Postupně si nyní projdeme každou desku zvlášť.

Hlavní deska

Hlavní deska se nachází v souboru *BoardMain*. Obsahuje vlastnosti odpovídající deskové podobě. Je zde reference na aktuálního pladina (nebo reference obsahuje *null,* pokud paladin přešel ke hráči). Dále zde můžeme vidět *listy* monster a místnosti, do kterých se vloží nová nabídka každé roční období. Nachází se zde i reference na obchody, které si popíšeme níže. Poslední vlastnosti je zadní část kouzelné karty, která zobrazuje, o který herní rok se aktuálně jedná (šedý meč pro první rok, druhý rok se vyznačuje dvěma zlatými meči). Metody jsou využity dvě. Jedna je *(clean),* která čistí

nabídky monster a místnosti. Tou druhou je metoda *showWinner* vyvolávající událost pro zjištění a vypsání vítěze, tato metoda bere *list* hráčů.

Jak jsem uvedl již výše, hlavní deska obsahuje referenci na nákupní pole typu *ShopingPark*. Tento datový typ se nachází v souboru *shop* společně se všemi herními obchody, které implementují rozhraní *IShop.* Ve třídě *ShopingPark* se nacházejí reference na všechny obchody (ty se v konstruktoru naplní instancemi odpovídajících tříd). Zobrazení na obrazovce probíhá pomocí události, jejíž způsob je vysvětlen výše. Jsou zde i dvě metody, první z nich jménem *UseShops,* přijímá jako parametry hlavní desku (kvůli nabídce monster a místnosti), balíček pasti a balíček kouzel. Tato metoda postupně vyhodnotí všechny obchody v pořadí, v jakém uvádějí pravidla hry. Druhou a zároveň poslední použitou metodou lze nastavit potřebným obchodům vlastnost, která uvede, že se jedná o druhý rok hry.

Rozhraní *IShop* přikazuje každému obchodu vytvořit šest metod. První dvě (*InserPlayer*) jsou přetížené, slouží totiž k vložení hráče do obchodu na základě jeho karty příkazů. V první nejčastější variantě přijímá hráče a vrací pravdivostní hodnotu a úspěch tohoto vložení. Druhá přetížená metoda nevrací pravdivostní hodnotu, ale za to přijímá, na jakou pozici hráče v obchodě vloží. Toto je využíváno na začátku při rozdávání karet příkazů u nehrajících hráčů. Další metoda *UseShop* vyvolá využití obchodu v pořadí, jaké specifikují pravidla hry. Poslední tři jsou pro využití každého boxu zvlášť a jsou volány pouze uvnitř tříd. Veškerá okna vyvolávají událost pro zobrazeni *AskWindow* z *gui* a některá specifická okna pro svůj obchod (nakupování monster, výběr pasti...).

Pro posílání informací o změně v obchodě se využívá třída *BuidPacked*, která se nachází také ve složce *shop*, ale k tomu se dostaneme dále.

Časová deska

Časová deska v sobě nese tři druhy informací. Pozici na časové desce, uspořádaní herních událostí a hrdiny (kteří jsou rozděleni do třech vlastností). Časová hodnota začíná na čísle 15 a končí na čísle 44, to kvůli indexu obrázku v *gui.* S tímto souvisí metoda *NextTime* jehož jedinou funkci je zvýšit pozici na časové ose. Jsou zde i čtyři metody, které spolu souvisí a pracují pouze spolu. Metoda *GetHeros* má za úkol rozdat hrdiny podle pozice na stupnici zla aktuálního hráče. Z tohoto důvodu přijímá *list* hráčů, pozici začínajícího a číslo ročního období (aby metoda věděla, jaké hrdiny má rozdávat). Na počátku zavolá metodu *makePlayer*, která je soukromá pro tuto třídu a jejímž úkolem je seřadit hráče podle pozice na stupnici zla. Nakonec dle specifické metody pro každé období se hrdinové rozdají hráčům. Další dvě metody *payMonster* a *payRooms* berou jako parametr hráče a vyvolají pro tohoto hráče událost, jejímž cílem je zobrazení okna. Samy tyto dvě metody jsou reakci na vyvolanou události ve třídě této herní události. Poslední metoda, která se zde vyskytuje je *AktualizeTime* sloužící při načtení hry k zobrazení aktuální pozice času na této desce.

Bojová deska

Tato deska je podstatně jednodušší než deska časová. Obsahuje informace o časové pozici v bojové části (tentokrát od 4 do 12) a *list* kouzel. Co se týče metod, jsou zde použity dvě. První je stejná jako *AktualizeTime* a plní také stejný účel. Druhá metoda se nazývá *ShowMagic.* Jako parametr bere číslo útoku. Dle tohoto čísla pozná, jakou kartu ze čtyř možných kouzel otočit lícem nahoru.

Deska hrajícího hráče

Tato deska je nejobsáhlejší, co se týče prvků umístěných na ní. Proto i v mé PC verzi nacházíme nejvíce vlastností. Tato třída se jmenuje *BoardPlay* a obsahuje vlastnosti popisující hodnost hráče. Pasti jsou rozděleny na pasti připravené k obraně a pasti uložené v zásobě. Monstra jsou rozdělená stejně jako pasti. Dále zde jsou celočíselné datové typy uchovávající hodnoty o množství zlata, jídla, chycených hrdinů, chycených monster (na položce vězení se tyto dvě informace sčítají), množství impů, úředníků a minusové body. Nacházíme zde reference na podzemní třídy *Undergrounds*, také je zde odkaz na paladina, který je prázdný pokud žádný na podzemí neútočí. Samozřejmě nemůže chybět *list* s hrdiny. Jako poslední jsem si nechal reference pro práci s kartami příkazů. Při vytváření této třídy se v konstruktoru vytvoří osm karet příkazů do *listu PackedCards*. Z tohoto seznamu se následně předávají karty do balíčku *Cards* (vybrány karty příkazů k požití) a *Unused_cards* (použity karty příkazu, nemožno použít aktuální roční období). V každé části hry, v každém ze tří balíčků je pouze jedna karta příkazů. K práci s těmito kartami se využívá *list* (*Unused_cards*) pravdivostních hodnot je celkově osm, každá pravdivostní hodnota odpovídá jedné kartě příkazů a pokud je aktivní znamená to, že karta byla vložena do balíčku *Cards*, ale nemohla být využitá (plné nákupní pole, rozhodnutí hráče...).

Z metod je zde *setHero*, která má jako parametr hrdinu, který se má složit k hrdinům, kteří na podzemí útočí. Vloží je buď na poslední místo, nebo na první pokud se jedná o bojovníka. Další je metoda sloužící k nalezení obranné části podzemí, samozřejmě využívá metodu *GetDefenceRoom* ze třídy *Undergrounds*. Dále je zde metoda, která má za úkol otočit past, která byla vybrána k obraně lícem nahoru. Poslední metoda slouží při načítání síťové hry, *SetPlayer* nastaví vlastníka této desky.

Deska nehrajícího hráče

Poslední deska nacházející se ve třídě *BoardNoPlay* slouží pro zobrazování karet příkazu nehrajícího hráče. Jsou zde dva balíčky *Cards,* pro aktuálně používané karty a *Unused_cards,* pro karty používané z minulého ročního období. O veškerou další funkcionalitu se stará třída nehrajícího hráče.

4.3.3 Složka se všemi hráči

Z herních desek nyní přejdeme k hráčům, kteří jsou uložení ve složce *players*. Je zde herní rozhraní, které musí implementovat každý hráč. Také tady jsou dvě dědičnosti od hrajícího hráče (*PlayersPlay*), zde dědí síťový hrající hráč (*NetworkPlayersPlay*). Stejný vztah je mezi nehrajícím hráčem (*PlayersNoPlay*) a síťovým nehrajícím hráčem (*NetworkPlayersNoPlayer*). Dědicové od otcovských tříd využívají metody, některé však přepisují, více o nich si povím při síťové komunikaci.

Je zde využívána třída *Pickup*, která už byla zmiňovaná výše, i když jsme ještě neznali její název. Slouží při obraně podzemí, jako nastaveni příznaku v boji, jestli se hrdinové budou léčit, jaké bylo použito kouzlo, nebo jestli bude použita speciální místnost k obraně atd....

Rozhraní hráčů

Rozhraní[6] *IPlayers* musí implementovat všichni hráči, i síťoví. Jsou zde základní metody pro zprávu hráčů, všechny jsou povinné, takže u nehrajících hráčů se stává, že některé z nich mají prázdné tělo metody. První metoda, na kterou bychom se podívali je *ChooseCards*, tato metoda se volá pro výběr karet příkazů pro aktuální období, u nehrajících hráčů se vyberou náhodně. V pořadí druhá metoda přijímá, jako parametr aktivní hru (třída *Game*) a jmenuje se *setEvent*, slouží k přiřazení obchodu hlavní desky, aby zde hráči mohli vkládat své úředníky. Potom zde je metoda (*getNice*) pro získání hodnoty hodnosti na stupnici zla. Další metoda (*evaluateCards*) slouží k otočení karty příkazů, která byla toto roční období vybrána, otáčení probíhá po jedné kartě, abychom splňovali pravidla hry. Další dvě metody slouží k nastavení a stažení impů do podzemí, jejich jména jsou *useImp* a *getImp*. Metoda jménem *changeCard* slouží k přetažení aktuálně použitých karet příkazu k použitým a stažení použitých karet. Následně je zde další metoda, která je u nehrajících hráčů prázdná, ale u hrajících slouží k nastavení obrany, před útokem hrdinů. *GetPosition*, getTypPlayer a *getColor* nám jako návratový typ vrátí pozici v kruhu, typ hráče a barvu hráče. Před začátkem hry musí dojit k výběru počátečních použitých karet příkazů a k tomu slouží metoda *startGame*. Pro stažení hrdinů slouží metoda *LeaveHeros*. Při nahrávání hry se volá metoda *LoadGame*, která vykreslí všechny prvky desky na obrazovku a bere jeden pravdivostní parametr, který určuje, zda se jedná o síťovou hru. Poslední dvě metody slouží k nastavení síťové hry a zjištění zda se jedná o síťovou hru (*SetNetworkGame, GetNetworkGame*).

Hrající hráč

Třída *PlayersPlay* představuje žijícího hráče sedícího za počítačem a slouží k ukládání informaci o něm a aktuálním stavu, v jakém se jeho deska nachází. Z toho vyplývá, že jedna z jeho vlastnosti musí být třída *BoardPlay*, dále také jméno, barva, typ hráče, pozice, počet ukázaných kouzelných karet tento rok. Jsou zde i informace doplňující aktuální stav hry a to, zda bylo na hráče použito

kouzlo *Grafiti*, reference na obchody, rok hry a zda se jedná o síťovou hru, pokud ano zda se jedná o server a případné odkazy na klienta a server.

K výše zmiňované metodě *evaluateCards,* je zde doplněna pomocná metoda *SetOfficials*, která se již přímo podle přijaté karty, jako parametr, pokusí poslat úředníka do obchodu. Další metoda, která není součástí rozhraní je *Defencing*, která přijímá aktuální kouzlo a číslo na útoku, který probíhá na podzemí. K obraně podzemí slouží pomocné metody, postupně se volají následovně: Nejdříve se zkontroluje, zda je ještě nutné se bránit, to znamená, pokud nejsou všichni hrdinové a paladin mrtví. Poté se nastaví parametry příznaků. Následně pomoci metody *useTraps*, se využije zvolená past, pak se podívá, zda se jedná o rychlé kouzlo, pokud ano provede se. Pomoci metody *useMonster* provedou obranu vybraná monstra. Jestliže je nyní aktivní pomalé kouzlo následuje léčení hrdinů za pomoci metody *healing* a poté dobití podzemí metodou *destroyUnderground,* pokud bylo použitá past *PoisonArrow*, tak bude dokončen její útok pomoci metody *finalizeTrap* a poté bude použito *finalize* sloužící k zobrazení aktuálních změn na hrací desce. Většina těchto metod přijímá balíček příznaků typu *Pickup*, aby byla jejich funkčnost odpovídající aktuálnímu stavu hry. V této části se využívají tři metody pro zjištění součtu speciálních vlastnosti hrdinů a to u zlodějů, kouzelníků a kněžů, tyto metody se jmenuji *getTiefAll, getMagicianAll, getPriestAll*.

Jsou zde i metody určené pro síťovou hru, ale o těch si povíme až v kapitole věnované této problematice.

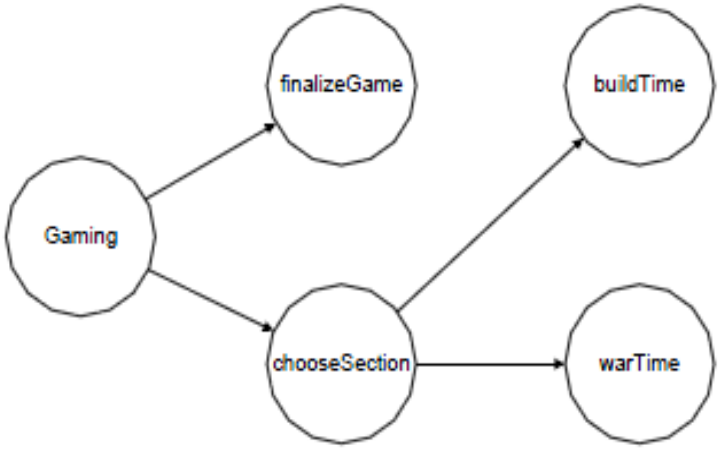
Nehrající hráč

Důležitou třídou při hře méně než tři hráčů, je třída *PlayersNoPlay*, ta představuje soubor informací o nehrajícím hráči. Tento typ hráče vlastní desky typu *BoardNoPlay* a doplňující informace, které hrající obsahuje ve třídě své desky, to je *list* karet příkazů, do kterého se na počátku vloží všech osm typu karet a také informaci uchovávající hodnost hráče. Dále je zde klasicky jméno, barva, typ hráče, typ hry (tuto informaci potřebujeme na rozdíl od hrajících hráčů, abychom věděli jak se zachovat při otáčení karet příkazu nahoru), reference na obchody, pozici, vlastnosti určující síťovou komunikaci.

Co se týká metody je zde také přidána metoda *SetOfficials*, která ale vkládá úředníky na druhou pozici. Stejnou funkčnost, jako u nehrajících hráčů má metoda *SetOfficials2*, ta se pokusí vložit úředníka na první volnou pozici.

4.3.4 Složka s řízením hry

V souboru *game* se nachází jediná třída a to třída *Game,* která řídí celou hru a rozhoduje o tom, co se bude dít. Tato třída pracuje se všemi ostatními řídícími prvky, které byly dosud uvedeny. Má dva konstruktory jeden pro lokální hru a druhý pro síťovou. Ať už se jedná o jakoukoli hru, v konstruktoru vytvoří instance tříd: *BoardMain, BoardTime,BoardWar, Monsters, Heros, Traps, Magics* a *Rooms*. A nově vytvořené instance přiřadí svým vlastnostem. Také v konstruktoru určí o jakou hru se bude jednat, tím je myšleno, kolik hrajících hráčů bude hrát.



Obr. 23: Znázornění rozhodování hry

Po vytvoření nové instance a provázání události *gui* zavolá metodu *StartGame*, která nastaví všechny řídicí vlastnosti na počátek a zavolá nad každým hráčem metodu *startGame.* Jako předposlední věc ještě propojí události pro řízení pohybu paladina mezi hráči a to v metodě *setPaladinEvent*. A nakonec iniciuje první rok hry, tato iniciace nám již zobrazí důležité informace pro rozhodování v prvním ročním období. Ještě bych se chtěl vrátit ke zmíněným řídícím vlastnostem, jsou tři: první z nich *Year* nám říká, v jakém roce se aktuálně nacházíme, druhou vlastnosti je *Section*, která nám říká, zda se jedná o budující nebo bojovou část hry a poslední je *Part*, určující jak v bojové tak i budovatelské jednu ze čtyř možných bodů, kdy se hra pozastaví.

Po stisknutí tlačítka pokračovat na hlavním okně, se hra ocitne v metodě *Gaming* a následně se podle výše uvedeného diagramu Obr. 23 rozhodne, co bude dál. Nejdříve se hra podívá, zda je konec hry, nebo je ještě v čem pokračovat, pokud ano podle řídicích proměnných se rozhodne[7], jestli bude hra pokračovat v bojovém nebo budujícím stavu. Po každém ukončení činnosti se stavy znovu aktualizují. Pokud se přijde na to, že skončila, znovu se zobrazí okno vítězů, které se vyvolá metodou *finalieGame.*

Rozhodl jsem se popsat řízení budující a bojové části postupně, nyní bych rád popsal budující část. Tato část hry je volána metodou *buildTime*, která jako první věc zkontroluje řídicí vlastnost *Part*, pokud se nejedná o poslední, zobrazí se další herní událost lícem vzhůru. Dále se zavolá metoda *makeCirkle,* tato metoda bere jako parametry hodnotu aktuálního roku a hodnotu v jaké budovací části hry se nacházíme. Na začátku této metody zavolá metodu *playrsTurn,* jejímž úkolem je získat od všech hráčů karty příkazu a vložit jejich úředníky do obchodů a následně použít obchody. Po návratu z *makeCirkle* se podíváme, jestli se nenacházíme v herní zimní části, pokud ano použijeme herní událost a rozdávání hrdinu, tak je pomoci časové desky rozdáme hráčům. Potom dojde k dokončení cyklu a vyzvednutí impů z herních plánu. Nakonec se nastaví další prvky pro příští roční období. Mezi těmito popsanými metodami dochází k posouvání času na na časové desce.

Nyní se pustíme do bojové části, ta je realizovaná v metodě *warTime*, která nejdříve zkontroluje, zda se jedná o první boj. Jestliže kontrola bude úspěšná, iniciuje boj pomoci metody *inicializeFight*, která vyvolá metodu pro zobrazení bojové desky do hlavního okna, poté zavolá metodu *makeFight*, která řídí aktuální útok hrdinů. Tato metoda pomocí volání *setDefence* nastaví všechny hrající hráče pro obranu. Poté nechá zobrazit kartu kouzla, a už nic nebrání útoku hrdinů pomoci metody *atackHero,* tato metoda bere dva parametry a to aktivní kouzlo a číslo útoku. V metodě *atackHero* se postupně vyvolá útok na každého hráče. Následně se aktualizující řídicí vlastnosti, a pokud se jedná o první rok a poslední útok, zavolá se metoda *inicializeYear*. Mezi úkoly se zase posouvá ukazatel času na bojové desce.

4.4 Síťová komunikace

Možnost hrát tuto hru po síti, je jedením z nejdůležitějších důvodů, proč hru převádět na PC verzi. Hra komunikuje mezí sebou tak, že zakladatel hry bude i zároveň serverem, a přijme odpovídající počet klientů podle nastavených počtu hráčů o jedno menší (první je sám server). Ve zkratce komunikace bude probíhat dvěma způsoby. První pokud aktivní hráč je serverem, pak po ukončení svého tahu rozešle své rozhodnutí i ostatním hráčům, ti už na odpověď čekají. Druhou možnosti je, když aktivní hráč je klient a ostatní klienti a server čekájí na odpověď, pak klient odešle zprávu serveru, ten jej zpracuje a odešle všem ostatním kromě původního odesilatele.

4.4.1 Server

Pokud se hráč rozhodne založit hru jako server, projekt *gui* vytvoří instanci třídy *Servr* z projektu s*erver* (jméno toho projektu je docela nepřesné, protože obsahuje i třídu *Client*). Konstruktor si vytvoří *listy* tříd *NetworkStream* a *TcpClient*, do nichž se později vloží informace o příchozích klientech. Dále je zde důležitý formát, v jakém se data odesílají a přijímají, ten je zde binární

(*BinaryFormatter*)[8]. Po zahájení poslouchání na zadaném portu přes instanci třídy *TcpListener* se vytvoří okno, které po kliknutí na tlačítko *Připojit* začne přijímat klienty, kteří se snaží připojit k serveru. Na příjem klientu je časový limit. Pokud do nastaveného času nepřijde žádný klient, hra nahlásí chybu. Při přijeti klienta se časovač restartuje. Příjem klientu je realizován v metodě reagující na událost stisku tlačítka *Připojit.* Po úspěšném přijetí požadovaného počtu klientu začíná práce se jmény.

Ve třídě *Servr* a metodě *getNames* můžeme poprvé vidět přijem zprávy od klientů, kteří posílají své jméno. Také se zde nastaví *Timeout* pro příchozí zprávy od každého klienta. Pokud do této doby nepřijde zpráva, hra nahlásí chybu a ukončí se. Dále dochází k práci se jmény, která už byla popisována. Po ukončení práce se jmény se pošle klientům informace o hře a to v podobě třídy *PlayerInfo*, tato třída obsahuje jména všech hráčů (po úpravě), počet hráčů a místo v „kruhu“ přijímacího hráče. Následně hra umístí jednoho hrajícího hráče na první pozici a místo všech ostatních hrajících hráčů vloží síťové hrající hráče, nehrající hráče vytvoří normálně. Nově vytvořené hře je následně na každé potřebné místo nastaveno, že se jedná o síťovou hru a také o server a přiřadí se jim reference na server. Tyto místa jsou ve třídách *Game* všech druhů hráčů a všech obchodů.

Při práci s příjmem a odesláním dat na serveru se ve hře využívají čtyři základní metody. První dvě z nich *sentToClient* jsou přetížené a slouží k odeslání objektu, který přijímají jako parametr, ve druhé variantě je přidán jeden parametr navíc a to číslo hráče, kterému se objekt posílat nemá. Metoda *setFromPlayer*, přijímá jako parametr číslo hráče, od kterého má přijmout paket. Poslední metoda je *waitToOthers* sloužící při čekání na potvrzení tlačítka *Pokračovat* od všech klientů, pokud přijme potvrzení od všech, rozešle se následně všem klientům informace o pokračování ve hře. Pokud se jeden z hráčů rozhodne ukončit hru, server rozešle tuto informaci ostatním klientům.

Při načítání síťové hry se postupuje podobně, využívá se však jiného okna pro připojování klientů a kontrolují se jména hráčů. Po správném připojení se jména všech hráčů znovu nezpracovávají. Server pošle klientům celou uloženou hru.

4.4.2 Klient

Druhou stranu komunikace samozřejmě zajišťuje klient, který je součástí každého hráče, není- li zakladatel. Budu vlastně nyní popisovat úplně opačný pohled na věc než tomu bylo u serveru. Potom co hráč nastaví povinné údaje, vytvoří se instance třídy *Client*, který pomoci IP adresy a portu vytvoří spojení přes třídu *TCPClient.* V konstruktoru se dále nastaví binární formátovaní a získá se vlastnost *NetworkStream* již připojeného klienta. Pokud na uvedené adrese a portu nikdo neposlouchá, program to oznámí a nechá vyplnit parametry znovu. Po úspěšném připojení k serveru hráč odešle své jméno. V následující části se rozhodne, jestli se hra připojuje k nahrané hře ze souboru, nebo byla nově založena.

Pokud klient obdrží pravdivostní hodnotu, ví, že se jedná o nahranou hru. V přijaté odpovědi je odpověď, zda se jeho jméno schoduje se jménem uloženého hráče. Pokud ne, komunikace se ukončí. V opačném případě hra následně obdrží celou uloženou hru. Prací klienta bude předělat odpovídajícího síťového hráče na nesíťového. Dále má za úkol předělat nehrající hráče na síťové nehrající hráče a původního hrajícího hráče na síťového hráče. Následně se provážou události, všechny hodnoty se zobrazí a hra je připravena k pokračování.

Pokud hráč obdrží místo pravdivostní hodnoty třídu typu *PlayerInfo* ví, že se jedná o nově založenou hru. Umísti svého hrajícího hráče podle přijatých dat a nastaví síťové hráče na všechny ostatní hrající hráče a nehrající hráči budou síťoví nehrající. Poté se na všechny potřebná místa, jako u serveru vloží informace, že se jedná o síťovou hru, že se nejedná o server a vloží se instance klienta.

Při komunikaci se serverem jsou používány tři metody ve třídě *Client*. První dvě z nich mají stejnou funkčnost, jedná z nich ale jako parametr přijímá celočíselnou hodnotu. Metody slouží k přijetí dat od serveru. U metody s číselným parametrem se jedná o čekání na stisk tlačítka *Pokračovat* ostatními hráči. Proto se musí nastavit větší *TimeOut.* Metoda pro odesílání dat se jmenuje *sentToServer*, tato metoda přijímá parametr, který odešle severu.

4.4.3 Síťový hrající hráč, síťový nehrající hráč a komunikační balík

Jak bylo uvedeno v kapitole o logice hry, projdeme si nyní dva zbývající hráče určené pro síťovou komunikaci. Obě tyto třídy dědí od svých hrajících hráčů. Některé metody rodičů jsou ale označeny jako virtuální a jsou u jejich síťových potomků přepsány. Většinou se jedná o změnu, která pouze čeká na přijaté změny od serveru, pokud se tedy jedná o klienta. Jde- li o server, nehrající hráč informaci přijme a rozešle ji ostatním klientům. Původní odesílatelé informací jsou tedy nesíťově hrající hráč a síťový hráč. Kteří se pokaždé na svém tahu podívají, zda se jedná o síťovou hru a pokud ano, rozešlou své pozměněné údaje podle toho, jestli se jedná o klienta nebo server.

Další třída, ke které bych se rád vrátil je *BuidPacked* u obchodů. Tato třída byla vytvořena pouze se záměrem komunikace při změnách v obchodech. Obsahuje vlastně veškeré data, která se mohla u hráče v obchoďe změnit. Při každém použití obchodu, se tato třída rozešle ostatním.

4.5 Společné datové typy

Projekt *typs* byl vytvořen za účelem sdílení datových typů mezi ostatními projekty. Je zde k nalezení pět specifických souborů, které byly logicky rozděleny podle smyslu.

První *board* obsahuje dva výčtové datové typy, tykajících se herních plánů. První z nich obsahuje barvy desek a druhý typ desky.

Největší soubor v tomto projektu, co se týče obsahu, se jmenuje *element. N*achází se v něm výčtové datové typy pro karty příkazů, rok kouzel (pro zobrazování správné zadní karty na hlavní desce), zda se jedná o rychlé či pomalé kouzlo, datový typ pro zobrazení zda hráč má aspoň jednu past, typ hrdiny, a jako poslední je zde šest výčtů pro správné indexování obrázku na všech herních plánech. Hrající hráč má dvě, přičemž druhá je k indexování popisku.

Výčtové typy pro řízení hry se nachází v souboru *game*. Jsou zde tři výčtové typy sloužící k identifikaci, v jaké části hry se zrovna hra nachází. Poslední je pro určení typu hry, tím se myslí počet hrajících hráčů.

V souboru (*player*), který je určený pro výčtové typy hráčů se nachází typ hráčů, který se navrací při zavolání metody *getTypPlayer* (předepisující *IPlayers*). Druhý datový typ je pozici v „kruhu“.

Poslední soubor uvedený v tomto projektu se jmenuje *PlayerInfo*, používaný k upřesnění klientům o jakou hru se jedná.

5 Testování a vyhodnocení

Součástí zadání a správného postupu při vývoji aplikaci je testování. Nejdříve pomoci základních testů jsem aplikaci testoval sám, a poté bylo mým úkolem sehnat dostatečný počet testerů. Tito testeři museli znát pravidla hry, aby mohli odhalit chyby v PC implementaci. Jejich úkolem bylo tedy hlavně hledat logické chyby v řešení, ale byl jsem rád za jakoukoliv připomínku. Při implementaci jsem strávil hodně času testováním jednotlivých částí, nejvíce s prvním rokem budování. Z tohoto důvodu v něm bylo nalezeno také nejméně chyb.

Při testování boje jsem si nejdříve volal metodu, která upravila hráče tak, že měli za sebou budovací část v prvním roce. Následně jsem mohl odhalit chyby, které měly za následek např. pád celé aplikace. Pro vyzkoušení druhého roku jsem simuloval celý herní rok, a tak jsem se dostal ke druhému roku bez zbytečné prodlevy. Po dokončení testování samotných části jsem přikročil k testování celé lokání hry a následně po implementaci síťové komunikace i k testování síťové hry.

Po tomto zkušebním testování jsem mohl začít s ostrým testem na dobrovolnících. Byly nalezeny některé logické a jedná závažná chyba. Tato chyba vedla k pádu programu. Veškeré tyto chyby byly úspěšně opraveny.

5.1 Průběh testování

Pro testování jsem vybral 11 lidí (včetně mě), kteří už hru hráli v deskové podobě. Těmto lidem bylo v rozmezí od 12 – 30 let. Každý z testerů měl jiné zkušenosti s hraním hry, a to z důvodu aby se případné chyby odhalily (zkušený hráč se soustředí na jiné věci než začátečník.) Každý z testerů provedl průměrně sedm zkušebních her. Některé hry vedly k odhalení chyby, jiné k úspěšnému dokončení hry (hlavně v pozdější fázi testování). Jelikož se jednalo o různě zkušené hráče, doba trvání hry byla také různá. Doba hrání hry se nicméně zkrátila díky PC, který se stará o potřebný chod hry. Samozřejmě kromě rozhodování hráčů. Doba hraní tedy z původní hodiny až dvou spadla na větší polovinu původního času. Po opakovaném hraní, tedy zvyknutí si testerů na GUI hru, se doba také zkracovala.

U některých hráčů, se objevila chyba, že aplikace postrádá *.Net Framework.* Těmto hráčům jsem doporučoval instalaci programu *.Net Framewokr 4.0*[1]. ve kterém byla hra vytvořena. Na starších verzích nelze zaručit stoprocentní funkčnost.

Celkově co se nalezených chyb týče, jednalo se povětšinou o chyby z nepozornosti. Mé počáteční testování jednotlivých částí, kterému jsem se hodně věnoval, bylo jistě velmi užitečné.

1 http://www.microsoft.com/downloads/cs-cz/details.aspx?FamilyID=9cfb2d51-5ff4-4491-b0e5-b386f32c0992

5.2 Vyhodnocení testování

Během testování byla odhalena řada chyb, které však byly následně úspěšně opraveny. V současné době si tedy nejsem vědom žádné chyby ve hře. Testeři celkově hodnotili hru kladně. Jediným problémem shledali zobrazování detailů. Tohoto problému jsem si byl již od počátečních návrhů vědom, a snažil jsem se ho řešit čím jak nejlépe. I přes tuto výtku shledávám hru jako hratelnou. Nedoporučuji však kvůli zobrazování detailů hraní na malých obrazovkách.

5.3 Možné vylepšení

Do budoucna by bylo určité dobré vymyslet jiný způsob zobrazování detailů, aby hra byla pro hráče přehlednější a příjemnější. Také zde nebyly implementovány karty události. PC verze je v této oblasti stejná jako neúplná verze deskové hry Vládci podzemí. Někteří hráči jsou touto absencí velmi potěšeni, někteří by však byli rádi, kdyby se o tyto karty události hra ještě rozšířila. Možnosti, zapnutí, či vypnutí těchto karet události by tedy bylo dobré do budoucna implementovat. Samotná desková hra při koupi obsahuje i rozšíření v podobě dalších prvků, i když přiložená pravidla neobsahují návod na jejich použití. Po nějaké době se však na internetu objevila aktualizovaná pravidla včetně tohoto rozšíření. Možnost hraní s rozšířením i na PC verzi je tedy také ke zvážení.

6 Závěr a zhodnocení

Cílem této bakalářské práce bylo seznámení se s deskovou hrou Vládci podzemí a také získaní veškerých podkladů pro implementaci této hry na počítač. Aplikace musí řídit průběh hry, hlídat pravidla a vytvořit jak lokání hru pro více hráčů na jednom PC, tak síťovou hru. Hru je možno ukládat a znovu nahrát dle volby uživatele.

Výsledkem úspěšné implementace jsou tedy čtyři projekty, z nich každý má specifickou funkci. Je to *gui, program, server, typs*. Řešení bylo rozděleno za účelem, snadného vyměnění části hry. Pokud se někdy bude zdát grafické uživatelské rozhraní (*gui*) jako nevyhovující, mohlo by být vytvořeno jiné a za stávající vyměněné. V takovém případě by se všechny ostatní projekty mohly zachovat. Vyměnit by se samozřejmě mohl jakýkoliv z projektu, ale musela by za něj být vytvořena odpovídající náhrada a také musela by splňovat odpovídající komunikační prostředky.

Velkým zklamáním při implementaci pro mě byla skutečnost, že jsem se nemohl držet navrhovaného zobrazování detailu hry, pomocí zoomovaného obdélníčku vedle kurzoru myši. Výsledná náhrada pomocí vyskakujících oken, by mohla byt doplněna i dalším vylepšením. Z časových důvodů jsem však toto vylepšení nyní nemohl provést. Toto zobrazování detailů bylo také občasnou výtkou při testování aplikace. Zobrazování detailů je do budoucna nejdůležitější věcí, kterou bude potřeba vylepšit.

Hra byla po úspěšném testování shledána, jako připravena k používání. Testování bylo prováděno na mladších lidech (12 – 30 let). U starších lidí jsem se již v minulosti setkal s jejich nechutí k učení se složitých pravidel, které tato hra obsahuje. Výsledná aplikace je tedy určená spíše pro mladší nadšence deskových her. PC implementace této hry zkracuje dobu hraní, a také nevyžaduje přítomnost hráčů na jednom konkrétním, stejném místě. Lze ji hrát také po síti. Dovoluje hráčům soustředit se jen na budování a obranu vlastního podzemí, logika PC hry se stará o veškeré nutné věci doprovázející tuto hru. V deskové verzi se o tyto doprovodné záležitosti museli starat sami hráči.

Literatura

[1] Chvátil, V.: *Vládci podzemí* [online]. prosinec 2009. Dostupný z WWW: <http://czechgames.com/files/DL_rulebook_CZ.pdf >

[2] Sharp, J.: *Microsoft Visual C# 2010 - Krok za krokem*. Brno, Computer Press, 2010. ISBN: 978-80-251-3147-3.

[3] Nash, T. : *C# 2010*. Brno, Computer Press, 2010. ISBN: 978-80-251-3034-6.

[4] Donahoo M. J., Calvert K. L.: *TCP/IP Sockets in C*. Waltham, Massachusetts, Morgan Kaufmann, 2009. ISBN-10 : 0123745403.

[5] Wróblewski P.: *Algoritmy*. Brno, Computer Press, říjen 2004. ISBN: 80-251-0347-9.

[6] Neward T. , Drayton P. , Albahari B.*: C# v kostce*. Praha, Grada., říjen 2003. ISBN: 80-247-0443-9.

[7] Töpfer P.*: Algoritmy a programovací techniky*. Praha, Computer Press, 2007. ISBN: 987-80-7196-350-9.

[8] switchonthecode.com: *C# Tutorial – Serialize Objects to File* [online]. Dostupný z WWW: <http://www.switchonthecode.com/tutorials/csharp-tutorial-serialize-objects-to-a-file>

Seznam příloh

Příloha 1. CD (zdrojové soubory, spouštěcí soubory)